The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

ASYNCHRONOUS MESSAGE SERVICE

DRAFT RECOMMENDED STANDARD

CCSDS 735.1-R-1

RED BOOK
February 2007

AUTHORITY

Issue:	Red Book, Issue 1
Date:	February 2007
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - The **standard** itself.
  - The anticipated date of initial operational capability.
  - The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING FOREWORD:)

As the computers used to conduct space flight mission operations both in flight and on the ground increase in capability, the software operating on those computers tends to increase in functional scope and thus take on greater operational significance. With that increase in scope comes increasing size and complexity, which may be partially mitigated by decomposition into modules whose functionality can be readily defined and tested. However, this modularity in turn entails a growing reliance on effective communication among modules. That is, mere decomposition cannot diminish the functional complexity implied by complex requirements. It can only partition that complexity into manageable portions, while the resulting web of communication relationships among modules introduces new complexity of a different order: rather than a relatively simple system of a few increasingly large and complex modules, a modern mission is increasingly likely to require a large and complex system of relatively simple modules.

Increasing complexity tends to increase the likelihood of failure. The increasing complexity of mission systems based on communication among modules therefore tends to increase the chance of such systems failing even as the success of those systems become increasingly critical to the achievement on mission objectives. Measures that can minimize the chance of failure in complex systems – exhaustive regression testing and configuration management, flight rules constraining the exercise of unproven system capabilities and the introduction of improvements – increase cost if they are taken and increase risk if they are not.

These considerations have led to the present recommendation for a standard system of communication – *messaging* – among mission software modules. The objective of this proposed messaging standard is to reduce mission cost and risk by confining much of the complexity of modern mission systems to relatively static and proven reusable infrastructure.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 735.1-R-1	Asynchronous Message Service, Draft Recommended Standard, Issue 1	February 2007	Current issue

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

CONTENTS

CONTENTS (continued)

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

This document defines a CCSDS Asynchronous Message Service (AMS) for mission data system communications. The service and its protocol implement an architectural concept under which the modules of mission systems may be designed as if they were to operate in isolation, each one producing and consuming mission information without explicit awareness of which other modules are currently operating. Communication relationships among such modules are self-configuring; this tends to minimize complexity in the development and operations of modular data systems.

A system built on this model is a 'society' of generally autonomous interoperating modules that may fluctuate freely over time in response to changing mission objectives, module functional upgrades, and recovery from individual module failure. The purpose of AMS, then, is to reduce mission cost and risk by providing standard, reusable infrastructure for the exchange of information among data system modules in a manner that is simple to use, highly automated, flexible, robust, scalable, and efficient.

This Recommended Standard specifies the protocol procedures and data units that accomplish automatic configuration of AMS communication relationships, dynamic reconfiguration of those relationships during operations, and the use of those relationships to accomplish the exchange of mission information among data system modules.

1.2 APPLICABILITY

This Recommended Standard specifies protocols and associated services that enable communication among modules of mission data systems, specifically:

– between modules of a ground data system;

– between modules of a flight data system;

– between modules of different ground data systems;

– between modules of the flight data systems of different spacecraft;

– between modules of flight data systems and modules of ground data systems, over interplanetary distances.

1.2.1 ORGANIZATION OF THE RECOMMENDED STANDARD

This Recommended Standard is organized as follows:

– Section 2 provides an overview of AMS, its intended use, and a description of the main interactions involved in message exchange.

– Section 3 defines the services provided by AMS along with the associated primitives and parameters.

– Section 4 defines the AMS protocol procedures.

– Section 5 defines the AMS protocol data units.

– Section 6 describes the AMS Management Information Base (MIB).

– Annex A provides a list of informative references.

– Annex B provides a list of acronyms and definitions.

– Annex C discusses the currently recognized underlying transport services for AMS.

– Annex D summarizes the functional significance of various classes of AMS subject numbers and the manner in which this significance affects protocol procedures.

1.3 CONVENTIONS AND DEFINITIONS

1.3.1 BIT NUMBERING CONVENTION

In this document, the following convention is used to identify each bit in an *N*-bit field. The first bit in the field to be transmitted (i.e., the most left justified when drawing a figure) is defined to be ‘Bit 0’; the following bit is defined to be ‘Bit 1’ and so on up to ‘Bit *N*–1’. When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, i.e., ‘Bit 0’.

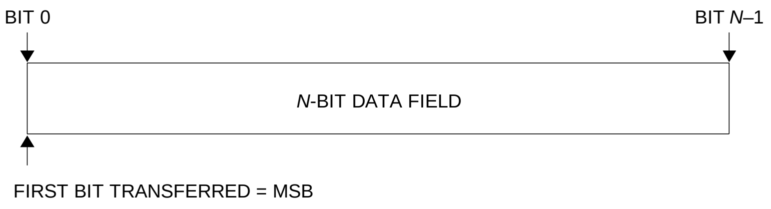

Figure 1-1: Bit Numbering Convention

In accordance with modern data communications practice, spacecraft data fields are often grouped into eight-bit ‘words’ which conform to the above convention. Throughout this Recommended Standard, the following nomenclature is used to describe this grouping:

8-BIT WORD = ‘OCTET’

Figure 1-2: Octet Convention

By CCSDS convention, all ‘spare’ or ‘unused’ bits shall be permanently set to value ‘zero’.

1.3.2 NOMENCLATURE

The following conventions apply throughout this Recommended Standard:

a) the words ‘shall’ and ‘must’ imply a binding and verifiable specification;

b) the word ‘should’ implies an optional, but desirable, specification;

c) the word ‘may’ implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

1.3.3 DEFINITIONS

1.3.3.1 Definitions from OSI Basic Reference Model

This Recommended Standard makes use of a number of terms defined in reference [1]. The use of those terms in this Recommended Standard shall be understood in a generic sense, i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

– entity;

– Protocol Data Unit (PDU);

– service;

– Service Access Point (SAP);

– Service Data Unit (SDU).

1.3.3.2 Definitions from Open Systems Interconnection (OSI) Service Definition Conventions

This Recommended Standard makes use of a number of terms defined in reference [2]. The use of those terms in this Recommended Standard shall be understood in a generic sense, i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

– Indication;

– Primitive;

– Request;

– Response.

1.3.3.3 Terms Defined in This Recommended Standard

Within the context of this document the following definitions apply:

A *continuum* is a closed set of entities that utilize AMS for purposes of communication among themselves. Each continuum is identified by a *continuum name* and corresponding non-negative continuum number. The continuum name that is the character string of length zero indicates 'all known continua' or 'any known continuum', whichever is less restrictive in the context in which this continuum name is used; the reserved continuum number zero corresponds to this continuum name.

An *application* is a data system implementation, typically taking the form of a set of source code text files, that relies on AMS procedures to accomplish its purposes. Each application is identified by an *application name*.

An *authority* is an administrative entity or persona that may have responsibility for the configuration and operation of an instance of an application. Each authority is identified by an *authority name*.

A *venture* is an instance of an application, i.e., a functioning projection of the application – for which some authority is responsible – onto a set of one or more running computers.

A *message* is an octet array of known size which, when copied from the memory of one module of a venture to that of another (*exchanged*), conveys information that can further the purposes of that venture.

The *content* of a message is the array of zero or more octets embedded in the message containing the specific information that the message conveys.

A *role* is some part of the functionality of an application. Each role is identified by a *role name* and corresponding non-negative role number. The role name that is the character string of length zero indicates 'all roles' or 'any role', whichever is less restrictive in the context in which the role name is used; the reserved role number zero corresponds to this role name. The role name 'RAMS' identifies Remote AMS (RAMS) gateway functionality as discussed below; the reserved role number '1' corresponds to this role name.

A *node* (i.e., a mission data system module) is a communicating entity that implements some part of the functionality of some AMS venture – that is, performs some application role – by, among other activities, exchanging messages with other nodes. Associated with each node is the name of the role it performs within the application. (Note that multiple nodes may perform the same role in an application, so the role name of a node need not uniquely identify the node within its venture.) In order to accomplish AMS message exchange a node generates AMS service requests and consumes AMS service indications; the node that is the origin of a given AMS service request or the destination of a given AMS service indication is termed *the operative node*.

A *message space* is the set of all of the nodes of one AMS venture that are also members of a given AMS continuum; that is, a message space is the intersection of a venture and a continuum. Each message space is uniquely identified within its continuum by the combination of the name of the application and the name of the authority that is responsible for the venture, and by a corresponding venture number greater than zero. (Note that unique naming of continua enables multiple message spaces that are in different continua but are identified by the same application and authority names to be concatenated via Remote AMS (discussed below) into a single venture.)

A *unit* (i.e., a unit of organization) is an identified subset of the organizational hierarchy of the nodes of one AMS venture, declared during venture configuration as specified by the responsible authority for that venture. Each unit is uniquely identified within the venture by *unit name* and corresponding non-negative unit number. The *root unit* of a venture is the unit that is coterminous with the venture itself; its unit name is the character string that is of length zero, and the reserved unit number zero corresponds to this unit name. A unit whose name is identical to the first N bytes – where N is greater than or equal to zero – of the name of another unit of the same venture is said to *contain* that other unit. The membership of a unit that is contained by another unit is a subset of the membership of the containing unit. Note that this means that units form a strict hierarchy: no node is ever a member of two units such that neither unit is a proper subset of the other.

A *cell* is the set of all nodes that are members of one unit of a given venture and are also members of a given continuum; that is, it is the intersection of a unit and a continuum. Since each unit is a subset of a venture, each cell is necessarily a subset of the message space for that venture in that continuum. Each cell is uniquely identified within its message space by its unit's name and number. The root cell of a message space is coterminous with the message space itself. A cell contains some other cell only if its unit contains that other cell's unit. A cell may be an empty set; that is, in a given continuum there may be no nodes that are members of the cell's unit. The *registered membership* of a cell is the set of all nodes in the cell that are not members of any other cell aside from cells which contain that cell.[1] (Note that the root cell contains every other cell in the message space, and every node in the message space is therefore a member – though not necessarily a registered member – of the root cell.)

The *domain* of an AMS service request is the set of nodes to which the request pertains. It comprises all of the nodes that are members of the venture in which the operative node is itself a member, with the following exceptions:

– If the service request is one for which continuum ID is not specified, then only nodes that are members of the local continuum are members of the service request's domain. Otherwise, if the service request's continuum ID parameter does not indicate 'all

[1] For example, if cell A contains cells B and C, and cell C contains cells D and E, any nodes in C that are not in either D or E are in the registered membership of cell C. Those nodes are also members of cell A, but because they are in cell C – which does not contain cell A – they are not in cell A's registered membership.

continua', then only nodes that are members of the continuum identified by the service request's continuum ID parameter are members of the service request's domain.

– Only nodes that are members of the organizational unit identified by the service request's unit ID parameter are members of the service request's domain.

– If the service request's role ID parameter does not indicate 'all roles', then only nodes performing the role identified by that role ID are members of the service request's domain.

The *subject number* (or *subject*) of a message is an integer embedded in the message that indicates the general nature of the information the message conveys, in the context of the AMS venture within which the message is exchanged. A *subject name* is a text string that serves as the symbolic representation of some subject number.

To *send* a message is to cause it to be copied to the memory of a specified node. To *publish* a message on a specified subject is to cause it to be sent to one or more implicitly specified nodes, namely, all those that have requested copies of all messages on the specified subject. To *announce* a message is to cause it to be sent to one or more implicitly specified nodes, namely, all those nodes that are located within a specified continuum (or all continua), that are members of a specified unit (possibly the root unit), and that perform a specified role in the application (possibly 'any role').

A *subscription* is a statement requesting that one copy of every message published on some specified subject by any node in the subscription's *domain* be sent to the subscribing node; the domain of a subscription is the domain of the AMS service request that established the subscription.

An *invitation* is a statement of the manner in which messages on some specified subject may be sent to the inviting node by nodes in the *domain* of the invitation; the invitation's domain is the domain of the AMS service request that established the invitation.

Some of the protocol procedures defined in this specification are in part expressed in terms of timeout intervals. All AMS timeout intervals are notional rather than fixed. (Note, however, that effective operation of the AMS protocols cannot be assured unless the same fixed values for all timeout intervals are used universally within any given set of interoperating AMS continua.) The timeout intervals referenced in this specification are defined as follows:

Table 1-1: AMS Timeout Intervals (in Seconds)

Interval Name	Interval ID	Nominal Value	Mandatory value
Configuration server response	N1	5	n/a
Registrar response	N2	5	n/a
Registrar heartbeat	N3	10	n/a
Node heartbeat	N4	n/a	2X the value of N3
Node loss imputation	N5	n/a	3X the value of N4

1.4 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommended Standard. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Standard are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS documents.

[1] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[2] *Information Technology—Open Systems Interconnection—Basic Reference Model—Conventions for the Definition of OSI Services*. International Standard, ISO/IEC 10731:1994. Geneva: ISO, 1994.

The latest issues of CCSDS documents may be obtained from the CCSDS Secretariat at the address indicated on page i.

2 OVERVIEW

2.1 GENERAL

2.1.1 ARCHITECTURAL CHARACTER

A data system based on AMS has the following characteristics:

– Any module may be introduced into the system at any time. That is, the order in which system modules commence operation is immaterial; a module never needs to establish an explicit *a priori* communication ‘connection’ or ‘channel’ to any other module in order to pass messages to it or receive messages from it.

– Any module may be removed from the system at any time without inhibiting the ability of any other module to continue sending and receiving messages. That is, the termination of any module, whether planned or unplanned, only causes the termination of other modules that have been specifically designed to terminate in this event.

– When a module must be upgraded to an improved version, it may be terminated and its replacement may be started at any time; there is no need to interrupt operations of the system as a whole.

– When the system as a whole must terminate, the order in which the system’s modules cease operation is immaterial.

AMS-based systems are highly robust, lacking any innate single point of failure and tolerant of unplanned module termination. At the same time, communication within an AMS-based system can be rapid and efficient:

– Messages are exchanged directly between modules (nodes) rather than through any central message dispatching nexus.

– Messages are automatically conveyed using the ‘best’ (typically – though not necessarily – the fastest) underlying transport service to which the sending and receiving modules both have access. For example, messages between two ground system modules running in different computers on a common LAN would likely be conveyed via TCP/IP, while messages between modules running on two flight processors connected to a common bus memory board might be conveyed via a shared-memory message queue.

Finally, AMS is designed to be highly scalable: partitioning message spaces into cells enables a venture to comprise hundreds or thousands of cooperating modules without significant impact on application performance.

2.1.2 MESSAGE EXCHANGE MODELS

AMS message exchange is fundamentally *asynchronous*. That is, each message is sent in a ‘postal’ rather than ‘telephonic’ manner: upon sending a message, an AMS node need not wait for arrival of any message (such as a *reply* to the message it sent) before continuing performance of its functions.

Although message exchange among nodes is asynchronous, for some purposes it may be desirable to apply the information in a reply message (received asynchronously) to the context in which the antecedent message was published. To this end, AMS enables a node to include a *context number* in any original (non-reply) message; AMS procedures can be used to reply to any original message, and the reply to a message automatically includes an echo of the context number (if any) embedded in the original message. This number can be used to retrieve some block of contextual information, enabling the original message sender to link the information in a reply message to the application activity which caused the antecedent message to be issued, so that this activity may be continued in a pseudo-synchronous fashion. The specific mechanism used to establish this linkage is an implementation matter.

For some purposes true message synchrony may be necessary as well; that is, it may be desirable for a node to *query* some other node, sending a message and then suspending operations altogether until a reply is received. AMS procedures additionally support this communication model when it is required.

Most message exchange in an AMS-based data system is conducted on the ‘publish-subscribe’ model:

– A node uses AMS procedures to announce that it is *subscribing* to messages on a specified subject.

– From that time on (until the subscription is canceled), whenever any node in the message space uses AMS procedures to *publish* a message on that subject, a copy of the message is automatically delivered to that subscribing node and to all others that have announced similar subscriptions.

This model can greatly simplify application development and integration. In effect, each node plugs itself into a data ‘grid’, much as producers and consumers of electric power – for example, a hydroelectric plant and a kitchen toaster – plug into an electric power grid. An AMS node can insert into such a data grid whatever data it produces, without having to know much about the consumer(s) of that data, and draw from the grid whatever data it requires without having to know much about the producer(s). The design of a node is largely decoupled from the designs of all other nodes in the same way that the design of a toaster is largely decoupled from the design of a power plant.

For some purposes, though, it may still be necessary for a node to send a message privately and explicitly to some specific node, e.g., in reply to a published message. Again, AMS procedures support this communication model as well when it is required.

2.2 ARCHITECTURAL ELEMENTS

2.2.1 GENERAL

The architectural elements involved in the asynchronous message service protocol are depicted in figure 2-1 and described below.

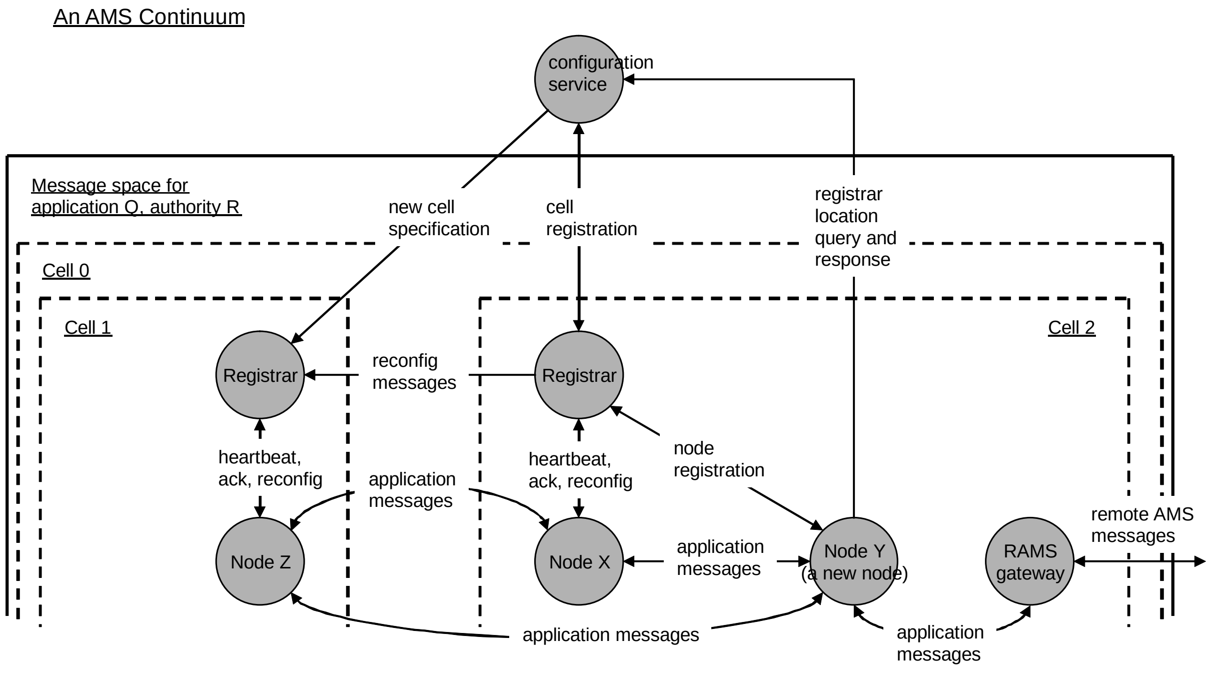

Figure 2-1: Architectural Elements of AMS

2.2.2 COMMUNICATING ENTITIES

All AMS communication is conducted among three types of communicating entities: nodes (defined earlier), registrars, and configuration servers.

A *registrar* is a communicating entity that catalogues information regarding the registered membership of a single cell of a message space. It responds to queries for this information, and it updates this information as changes are announced.

A *configuration server* is a communicating entity that manages a database of configuration information for a single continuum. In particular, it catalogues information regarding the message spaces that the continuum comprises, notably the locations of all registrars. It responds to queries for this information, and it updates this information as changes are announced.

2.3 OVERVIEW OF INTERACTIONS

2.3.1 TRANSPORT SERVICES FOR APPLICATION MESSAGES

AMS occupies a position in the OSI protocol stack model somewhere between level 4 (Transport) and level 7 (Application); AMS might best be characterized as a messaging 'middleware' protocol. As such, it relies on the capabilities of underlying Transport-layer protocols to accomplish the actual copying of a message from the memory of the sending node to the memory of the receiving node. It additionally relies on those capabilities to accomplish the transmission of the *meta-AMS* (or *MAMS*) messages to and from registrars and configuration servers that enable the dynamic self-configuration of AMS message spaces.

For any given AMS continuum, some common transport service must be utilized for MAMS traffic by all communicating entities involved in the operations of all message spaces in the continuum. The transport service selected for this purpose is termed the continuum's *Primary Transport Service* or *PTS*. Selection of the PTS for a continuum is an implementation matter; the Bundle Protocol of the Delay-Tolerant Networking architecture is a plausible choice, but for some continua an implementation based on UDP/IP may be more appropriate.

The PTS clearly can also be used for application message exchange among all nodes in a continuum, as it must be universally available for MAMS message exchange. In some cases, however, improved application performance can be achieved by using a different transport service for message exchange between nodes that share access to some especially convenient communication medium, such as a shared-memory message queue. These performance-optimizing transport services are termed *Supplementary Transport Services* or *STSs*.

The network location at which a node can receive messages via a given transport service is termed the node's *delivery point* for that service. Multiple delivery points (for different transport services) may be characterized by the same *service mode* – that is, by the same levels of transmission order preservation and diligence in message delivery. For a given service mode, the list of all delivery points that provide that mode of service to a given node – in descending order of preference (generally, descending order of network performance) – is termed the *delivery vector* for that service mode, for that node.

2.3.2 REGISTRAR REGISTRATION

Every message space always comprises at least one cell (the root cell), and each node resides within (is registered in) some cell; in the simplest case all nodes of the message space reside in the root cell. Each cell is served by a single registrar, which is responsible for monitoring the health of all nodes in the cell's registered membership and for propagating six kinds of message space configuration changes: node registrations and terminations, subscription and invitation assertions, and subscription and invitation cancellations. On reception of one of these reconfiguration messages from a node in its own cell's registered membership, the registrar immediately propagates the message to the other nodes in that registered membership and then to the registrars of other cells in the message space, as appropriate; on receiving such a message from a remote cell’s registrar, the registrar propagates it to the nodes in its own cell’s registered membership.

The registrars themselves register with the configuration server for the continuum within which the message space is contained. A list of all possible network locations for the configuration server, in order of descending preference, must be ‘well known’ – that is, included in the AMS Management Information Bases (MIBs) exposed to all registrars for all message spaces in the continuum – and each continuum must have a configuration server in operation at one of those locations at all times in order to enable registrars and nodes to register. (The manner in which this latter requirement is satisfied is an implementation matter.)

All registrars and nodes of the same message space must register through the same configuration server. The registrars and nodes for any number of different message spaces may register with the same configuration server.

2.3.3 NODE REGISTRATION

Each node has a single associated *meta-AMS delivery point (MADP)* at which the node is prepared to receive MAMS messages. A new node joins a message space by registering itself within some cell of the message space, i.e., by announcing its role name and its MADP to the cell’s registrar and thereby inserting itself into the cell’s registered membership. However, knowledge of how to communicate with that registrar cannot be hard-coded into the node because the relevant registrar might be running at different network locations at different times.

For this reason, the first step in registering a new node is to contact the configuration server at one of its well-known possible network locations; as with the registrars, a list of all possible network locations for the configuration server, in order of descending preference, must be included in the AMS MIBs exposed to all application nodes. The configuration server tells the new node how to contact its registrar. The node obtains a *node number* (which uniquely identifies it within the cell’s registered membership) from the registrar and thereby registers. The registrar ensures that all other nodes in the message space learn the new node’s role name, node number, and MADP. Those nodes in turn announce their own role names, node numbers, and MADPs to the new node.

2.3.4 MONITORING NODE HEALTH

In order to acquire and maintain accurate knowledge of the configuration of a message space (for application purposes, and also to avoid wasting resources on attempts to send messages to nonexistent nodes or per concluded subscriptions), it is important for each registrar always to detect the terminations of nodes in its cell's registered membership. When a node terminates under application control it automatically notifies its registrar that it is stopping. However, if a node crashes – or the host on which a node resides is simply powered off or rebooted – no such notification is transmitted to the registrar. For this reason, every node automatically sends a 'heartbeat' message to its registrar every N4 seconds. The registrar interprets three successive missing heartbeats as an indication that the node has terminated.

Whenever it detects the termination of a node (either an overt termination or a termination imputed from heartbeat failure), the registrar informs all other nodes in the cell's registered membership – and, via other registrars, all other nodes in the message space – of the node's demise.

When the termination is imputed from a heartbeat failure, the registrar also tries to send a message to the terminated node telling it that it has been presumed dead; if this node is in fact still running (perhaps it had merely hung trying to write on a blocked file descriptor), it terminates immediately on reception of this message. This minimizes system confusion resulting from other application behavior that may have been triggered by the imputed termination.

2.3.5 MONITORING REGISTRAR HEALTH

In addition to monitoring the heartbeats of all nodes in its cell's registered membership, each registrar issues its own heartbeats to those nodes on the same cycle. Each node interprets three successive missing registrar heartbeats as an indication that the registrar itself has crashed. On detecting a registrar crash, the node presumes that the registrar has been restarted since it crashed; it re-queries the configuration server to determine the new network location of the registrar and resumes exchanging heartbeats.

This presumption is reasonable because the reciprocal heartbeat monitoring relationship between a registrar and its nodes is replicated in the relationship between the configuration server and all registrars, but on a slightly shorter cycle. The configuration server interprets three successive missing registrar heartbeats as an indication that the registrar has crashed; on detecting such a crash it automatically takes some implementation-specific action to cause the registrar to be restarted, possibly on a different host, so by the time the registrar's nodes detect its demise it should already be running again.

Since the node heartbeat interval is N4 seconds, within the first N5 seconds after restart the registrar will have received heartbeat messages from every node in the cell's registered membership that is still running and will therefore know accurately the configuration of the cell. This accurate configuration information must be delivered to new nodes at the time they start up (so that they in turn are qualified to orient a newly restarted registrar to the cell's configuration in the event that the registrar crashes). For this reason, during the first N5 seconds after the registrar starts it accepts only MAMS messages from nodes that are already registered in the cell (i.e., have been assigned node numbers); if it accepted and processed a registration message from a new node before being certain of the status of all old ones, it would run the risk of delivering incorrect information to the new node.

2.3.6 CONFIGURATION SERVICE FAIL-OVER

It is of course also possible for a configuration server to be killed (or for its host to be rebooted, etc.). Each registrar interprets three successive missing configuration server heartbeats as an indication that the configuration server has crashed. On detecting such a crash, the registrar begins cycling through all of the well-known possible network locations for the continuum's configuration server, trying to re-establish communication after the server's restart, possibly at an alternate network location. While it is doing so it is not issuing heartbeats or responding to other messages, so eventually all nodes may infer that their registrars have crashed and therefore begin re-querying the now-dead configuration server to re-establish communication with their registrars; when the configuration server fails to respond, the nodes too will begin cycling through network locations seeking a restarted configuration server. While they are doing so, they too will not be responding to MAMS messages, so all message space dynamic self-configuration activity will eventually come to a halt. (Note, however, that application message exchange per the current message space configuration in effect at the time of the discovery infrastructure failure need not be affected.)

When the configuration server is restarted at one of its well-known possible network locations, however, all registrars will eventually find it and re-announce themselves to it, so that when application nodes finally find it they can re-establish communication with their registrars; all message space dynamic self-configuration activity will thereupon resume.

It is possible, in this sort of failure scenario, that multiple configuration servers may be operating concurrently for a brief time; for example, the perceived failure of a configuration server might have been caused by a transient network connectivity failure rather than an actual server crash. To resolve this sort of situation, each running configuration server periodically sends an 'I am running' MAMS message to every lower-ranking configuration server network location in the well-known list of configuration server locations. When a configuration server receives such a message, it immediately terminates; all registrars and nodes that were communicating with it will detect its disappearance and search again for the highest-ranking reachable configuration server, eventually bringing the continuum back to orderly operations.

2.3.7 CONFIGURATION RESYNC

Finally, every registrar can optionally be configured to re-advertise to the entire message space the detailed configuration of its cell's registered membership (all active nodes, all subscriptions and invitations) at some user-specified frequency, e.g., once per minute. This capability is referred to as *configuration resync*. Configuration resync of course generates additional message traffic, and it may be unnecessary in extremely simple or extremely stable operating environments. But it does ensure that every change in message space configuration will eventually be propagated to every node in the message space, even if some MAMS messages are lost and even if an arbitrary number of registrars had crashed at the time the change occurred.

Taken together, these measures make AMS applications relatively fault tolerant:

– When a node crashes, its registrar detects the loss of heartbeat within three heartbeat intervals and notifies the rest of the message space. Application message transmission everywhere is unaffected.

– When a registrar crashes, its configuration server detects the loss of heartbeat within three heartbeat intervals and takes action to restart the registrar. During the time that the cell has no registrar, transmission of application messages among nodes of the message space is unaffected, but the heartbeat failures of crashed nodes are not detected and reconfiguration messages originating in the cell’s registered membership (registrations, terminations, subscription and invitation assertions, and subscription and invitation cancellations) are not propagated to any nodes. However, after the registrar is restarted it will eventually detect the losses of heartbeat from all crashed nodes and will issue obituaries to the message space; and if configuration resync is enabled it will eventually re-propagate the lost reconfiguration messages.

– When a configuration server crashes, all dynamic self-configuration activity may eventually come to a standstill. But no application nodes fail (at least, not because of communication failure), and on restart of the configuration server all self-configuration activity eventually resumes.

2.3.8 SECURITY

AMS can be configured to confine service access to application modules that can prove they are authorized to participate. For this purpose, asymmetric MAMS encryption may be used as follows:

– The AMS MIB exposed to the configuration server contains a list of all applications for which registration service may be offered, identified by application name. Associated with each application name is the AMS public encryption key for that application.

– The AMS MIB exposed to every registrar in each message space contains a list of all functional role names defined for the message space’s application; this list limits the role names under which modules may register in that message space. Associated with each role name is the AMS public encryption key for the application node(s) that may register in that role.

– The AMS MIBs exposed to all registrars and application nodes in the message space contain the AMS public encryption key of the configuration server.

– The AMS MIBs exposed to the configuration server and to all registrars and application nodes in the message space contain the private encryption keys that are relevant to those entities.

As described later, this information is used to authenticate registrar registration and exclude spurious registrars from the message space, to authenticate node registration attempts and deny registration to unauthorized application modules, and to assure the authenticity, confidentiality, and integrity of MAMS traffic exchanged between nodes and their registrars.

In addition, the confidentiality and integrity of AMS message exchange may be protected at subject granularity. The AMS MIB exposed to each node of a given message space may contain, for any subset of the message subjects (identified by name and number) used in the message space's application:

– a list of the role names of all nodes that are authorized senders of messages on this subject;

– a list of the role names of all nodes that are authorized receivers of messages on this subject;

– encryption parameters, including a symmetric encryption key, enabling encryption of messages on this subject.

This information may be used to support secure transmission of messages on selected subjects.

The structure of security information elements and the manner in which MIBs are populated with these elements are implementation matters.

2.3.9 SUBJECT CATALOG

The structure of the content of messages on a given subject is application-specific; message content structure is not defined by the AMS protocol. However, the AMS MIB exposed to all nodes of a given message space will contain, for each message subject (identified by name and number) used in the message space:

– a description of this message subject, discussing the semantics of this type of message;

– a detailed specification of the structure of the content of messages on this subject;

– optionally, a specification of the manner in which a correctly assembled message is *marshaled* (that is, converted from processor-internal representation to an array of octets) for network transmission in a platform-neutral manner and, on reception, un-marshaled into a format that is suitable for processing by the application.

When AMS is requested to send a message on a given subject, the message content that is presented for transmission is always in a format that is suitable for processing by the application. In the event that this format is not suitable for network transmission in a platform-neutral manner, as indicated by the presence in the MIB of a marshaling specification for this subject, AMS will marshal the message content as required before transmitting the message.

When AMS receives a message on a subject for which a marshaling specification is present in the MIB, AMS will un-marshal the message content into a format that is suitable for processing by the application before delivering the message.

The structure of subject catalog information elements, the manner in which MIBs are populated with these elements (e.g., by linkage to an external information model), and the manner in which message contents are marshaled and un-marshaled are implementation matters.

Message subjects, as noted earlier, are integers with application-defined semantics. This minimizes the cost of including subject information (in effect, message type) in every message, and it can make processing in an AMS implementation simpler and faster: subscription and invitation information may be recorded in (possibly sparse) arrays that are indexed by subject number.

This implementation choice, however, would require that message management control arrays be large enough to accommodate the largest subject numbers used in the application. The use of extremely large subject numbers would therefore cause these arrays to consume huge amounts of memory. In general, it is best for an AMS application to use the **smallest** subject numbers possible, starting with '1'. AMS MIB developers should bear this in mind.

2.3.10 REMOTE AMS MESSAGE EXCHANGE

Because issuance of an asynchronous message need not suspend the operation of the issuing node until a response is received, AMS's message exchange model enables a high degree of concurrency in the operations of data system modules; it also largely insulates applications from variations in signal propagation time between points in the AMS continuum.

However, some critical MAMS communication is unavoidably synchronous in nature: in particular, a newly registering node must wait for responses from the configuration server and the registrar before proceeding with application activity. For this reason, the core AMS protocol is most suitable for use in operational contexts characterized by generally uninterrupted network connectivity and relatively small and predictable signal propagation times, such as the Internet or a stand-alone local area network. It is usually advantageous for all entities of any single AMS continuum to be running within one such 'low-latency' network.

AMS application messages may readily be exchanged between nodes in different AMS continua, however, by means of the *Remote AMS* (*RAMS*) procedures. RAMS procedures are executed by special-purpose application nodes called *RAMS gateways*. Each RAMS gateway has interfaces to two communication environments: the AMS message space it serves and the *RAMS network* – the mesh or tree of mutually aware RAMS gateways – that enables AMS messages produced in one message space to be forwarded to other message spaces of the same venture. RAMS gateways operate as follows:

– Each RAMS gateway opens persistent RAMS network communication channels to the RAMS gateways of other message spaces for the same venture, in other continua. The interconnected RAMS gateways use these communication channels to forward message petition assertions and cancellations among themselves.

– Each RAMS gateway subscribes locally to all subjects that are of interest in any of the linked message spaces.

– On receiving its copy of a message on any of these subjects, the RAMS gateway node uses the RAMS network to forward the message to every other RAMS gateway whose message space contains at least one node that has subscribed to messages on that subject.

– On receiving a message via the RAMS network from some other RAMS gateway, the RAMS gateway node forwards the message to all subscribers in its own message space.

RAMS operations generalize the AMS architecture as shown in figure 2-2 below:

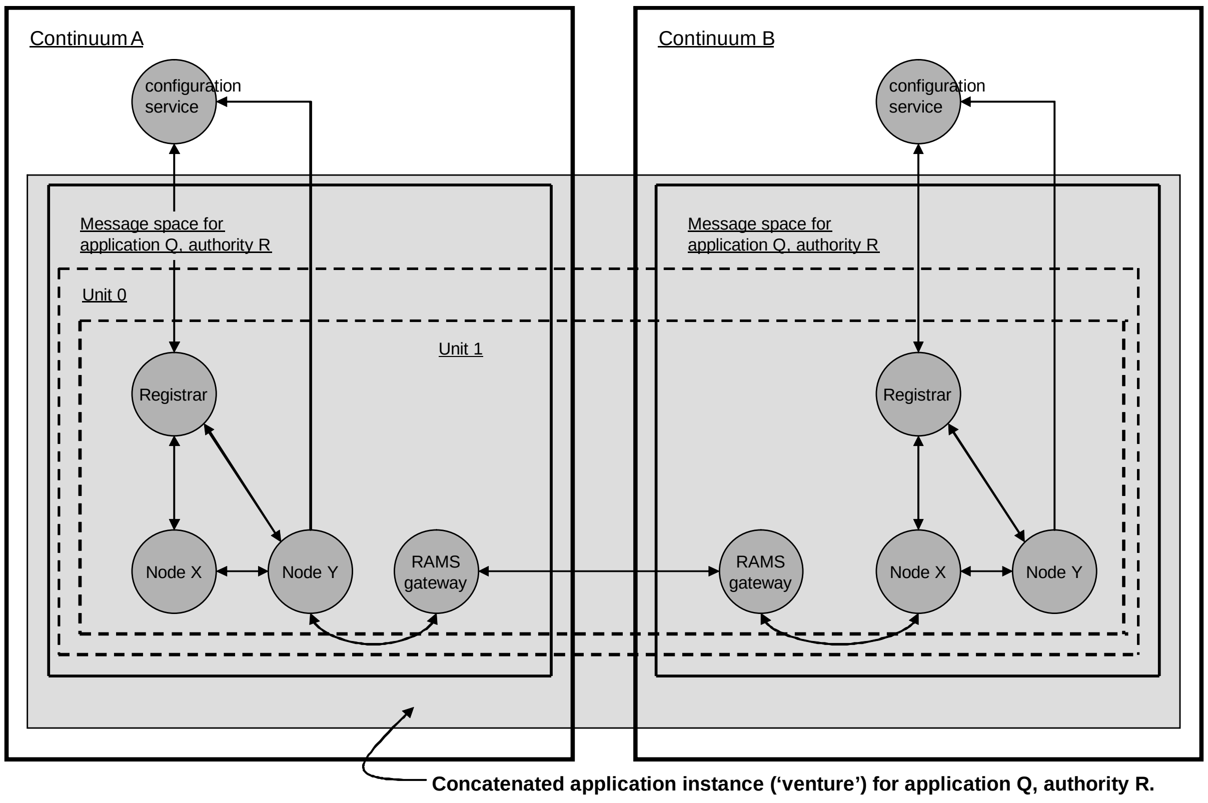
Continuum A
Continuum B
configuration service
configuration service
Message space for application Q, authority R
Message space for application Q, authority R
Unit 0
Unit 1
Registrar
Registrar
Node X
Node Y
RAMS gateway
RAMS gateway
Node X
Node Y
Concatenated application instance ('venture') for application Q, authority R.

Figure 2-2: General Remote AMS Architecture

In this way the RAMS protocol enables the free flow of published application messages across arbitrarily long deep space links while protecting efficient utilization of those links: only a single copy of any message is ever transmitted on any RAMS network communication channel, no matter how many subscribers will receive copies when the message reaches its destination message space.

Note that the nature of the RAMS network communication channels depends upon the implementation of the RAMS network. In order to communicate over the RAMS network for a given venture, each RAMS gateway must know the *RAMS network location* – expressed as an endpoint in the protocol used to implement the RAMS network – at which every neighboring RAMS gateway for that venture receives RAMS network traffic. Additional Management Information Base elements may be required for this purpose, but the configuration of RAMS networks is an implementation matter that is beyond the scope of this Recommended Standard.

Again, this extension of the publish/subscribe model to interplanetary communications is invisible to application nodes. Application functionality is unaffected by these details of network configuration, and the only effects on behavior are those that are intrinsic to variability in message propagation latency.

3 SERVICE DESCRIPTIONS

3.1 SERVICES PROVIDED TO THE APPLICATION

3.1.1 SUMMARY OF PRIMITIVES

The AMS service shall consume the following request primitives:

a) **Register.request**;

b) **Unregister.request**;

c) **Assert_invitation.request**;

d) **Cancel_invitation.request**;

e) **Assert_subscription.request**;

f) **Cancel_subscription.request**;

g) **Publish.request**;

h) **Send.request**;

i) **Query.request**;

j) **Reply.request**;

k) **Announce.request.**

The AMS service shall deliver the following indication primitives:

a) **Message.indication**;

b) **Reply.indication**;

c) **Fault.indication**;

d) **Register.indication**;

e) **Unregister.indication**;

f) **Assert_invitation.indication**;

g) **Cancel_invitation.indication**;

h) **Assert_subscription.indication**;

i) **Cancel_subscription.indication**.

3.1.2 SERVICE PRIMITIVE PARAMETERS

NOTE – The availability and use of parameters for each primitive are indicated in the definitions of primitives below, where parameters that are optional are identified with square brackets [thus]. The following parameter definitions apply.

3.1.2.1 The *continuum ID* parameter (name or number) shall identify the continuum (or continua) of AMS service to which the service primitive pertains. There shall be a one-to-one correspondence between AMS continua and authoritative AMS configuration servers. Continuum ID is required for Remote AMS (RAMS) service operations, where it enables messages passed between continua to be correctly dispatched.

3.1.2.2 The *application name* parameter shall identify the application served by a message space's venture.

3.1.2.3 The *authority name* parameter shall identify the administrative entity or persona that is responsible for a message space's venture. The combination of application name and authority name shall uniquely identify a message space within its continuum.

3.1.2.4 The *unit ID* parameter (name or number) shall identify some identified subset of the organizational hierarchy of the nodes in a message space's venture.

3.1.2.5 The *role ID* parameter (name or number) shall indicate the functional nature (role performed within the application) of a node.

3.1.2.6 The *Meta-AMS delivery point (MADP) specification* parameter shall be a *transport service endpoint specification* characterizing the manner in which a node is prepared to receive MAMS messages. The MADP specification shall identify a functional endpoint of the AMS continuum's primary transport service. The syntax in which a transport service endpoint specification is represented is transport service-specific; definitions of valid endpoint specification syntax for all recognized transport services are given in annex C.

3.1.2.7 The *node number* parameter shall uniquely identify a node within the cell in which it is registered.

3.1.2.8 The *subject ID* (name or number) parameter shall indicate the general nature of the application data in a message.

3.1.2.9 The *delivery specification* parameter shall characterize the manner in which a node is prepared to receive AMS messages. The delivery specification shall comprise a list of one or more *delivery point specifications*. Delivery point specifications shall be listed in declining order of preference; that is, the delivery point on which the node most prefers to receive messages shall be specified first, followed by the next-most-preferred delivery point, and so on. Each delivery point specification shall associate a *transport service name* with a transport service endpoint specification as described earlier. The transport service name shall be the name of the AMS continuum's primary transport service or the name of one of the continuum's supplementary transport services. By the nature of the indicated transport service, the *service mode* on which all messages addressed to a given delivery point are issued is implicitly associated with that delivery point. The service mode shall indicate (a) the *sequence* in which messages are delivered (Transmission order – nominally implying that message reordering upon arrival is performed automatically as necessary – or Arrival order) and (b) the *diligence* with which delivery of messages on this subject is attempted (Assured – nominally implying automatic acknowledgement and retransmission as necessary – or Best-effort).

3.1.2.10 The *service mode* parameter shall indicate the service mode governing issuance of messages on a given subject.

3.1.2.11 The *priority* parameter shall indicate the relative urgency of messages issued on a given subject. Priority shall be an integer in the range 1 to 15, where 1 signifies the greatest urgency and 15 the least.

3.1.2.12 The *flow label* parameter shall identify the message stream or ‘flow’ to which messages on a given subject contribute. Flow label is opaque to AMS but when passed through to transport services may be used to invoke additional, managed quality-of-service features. Flow label shall be an integer in the range 0 to 255.

3.1.2.13 The *context* parameter shall be an integer that identifies the application context in which a message was sent, if any. The significance and interpretation of this number are implementation matters. Conceptually, the context parameter functions within AMS as a message identifier. The context in which a reply is sent must be identical to the context in which the reply’s antecedent message was sent.

3.1.2.14 The *content length* parameter shall indicate the length (in octets) of the *content* parameter. The maximum value of this parameter shall be 65000.

3.1.2.15 The *content* parameter shall be an array of 0 – 65000 octets comprising the application data in a message, in a format that is suitable for processing by the application.

3.1.2.16 The *term* parameter shall indicate the length of time following message transmission within which a reply message should be received. If the term expires before the reply is received, the query has ‘timed out’.

3.1.2.17 The *fault expression* parameter shall indicate the nature of an operational fault encountered by AMS. The syntax of fault expressions is an implementation matter.

3.1.3 REGISTER.REQUEST

3.1.3.1 Function

The **Register.request** primitive shall be used by the node to commence its participation in a message space.

3.1.3.2 Semantics

Register.request shall provide parameters as follows:

```
Register.request (application name,
                  authority name,
                  unit ID,
                  role ID,
                  [MADP specification],
                  [delivery specification])
```

3.1.3.3 When Generated

Register.request is generated by the node at any time while the node is not currently participating in its message space.

3.1.3.4 Effect on Reception

Reception of **Register.request** shall, if approved, cause AMS to add the node to the indicated message space cell.

3.1.3.5 Additional Comments

If MADP specification is omitted, a default MADP specification based on preferences noted in the AMS MIB shall be computed. If delivery specification is omitted, a default delivery specification based on the standard delivery preferences noted in the node's AMS MIB shall be computed.

3.1.4 UNREGISTER.REQUEST

3.1.4.1 Function

The **Unregister.request** primitive shall be used by the node to terminate its participation in a message space.

3.1.4.2 Semantics

Unregister.request shall provide parameters as follows:

Unregister.request (node number)

3.1.4.3 When Generated

Unregister.request is generated by the node at any time while the node is participating in its message space.

3.1.4.4 Effect on Reception

Reception of **Unregister.request** shall cause AMS to remove the node from the indicated message space cell.

3.1.4.5 Additional Comments

The node number provided in the **Unregister.request** primitive must be the one that was provided in the **Register.indication** primitive that notified the node of its own successful registration.[1]

[1] The manner in which AMS service indications are linked to the AMS service requests to which they are directly responsive, where applicable, is an implementation matter.

3.1.5 ASSERT_INVITATION.REQUEST

3.1.5.1 Function

The **Assert_invitation.request** primitive shall be used by the node to indicate the manner in which private (that is, non-published) messages on a specific subject may be delivered to the node.

3.1.5.2 Semantics

Assert_invitation.request shall provide parameters as follows:

Assert_invitation.request (subject ID,
service mode,
priority,
flow label,
unit ID of invitee(s),
role ID of invitee(s))

3.1.5.3 When Generated

Assert_invitation.request is generated by the node at any time while the node is participating in its message space.

3.1.5.4 Effect on Reception

Reception of **Assert_invitation.request** shall, if approved, cause AMS to notify all nodes in the message space of the node's willingness to accept private messages on the indicated subject from nodes in the domain of the service request.

3.1.5.5 Additional Comments

The priority and flow label specified in the **Assert_invitation.request** are advisory, expressing the node's preferences for delivery of messages on this subject. Note that invitations are distinct from subscriptions: they enable private transmission of messages (*send*, *reply*, *query*, *announce*) on the indicated subject and **do not** mandate delivery of a copy of each published message on this subject.

3.1.6 CANCEL_INVITATION.REQUEST

3.1.6.1 Function

The **Cancel_invitation.request** primitive shall be used by the node to indicate that private messages on a specific subject may no longer be delivered to the node.

3.1.6.2 Semantics

Cancel_invitation.request shall provide parameters as follows:

```
    Cancel_invitation.request    (subject ID,
                                  unit ID of invitee(s),
                                  role ID of invitee(s))
```

3.1.6.3 When Generated

Cancel_invitation.request is generated by the node at any time while the node is participating in its message space.

3.1.6.4 Effect on Reception

Reception of **Cancel_invitation.request** shall cause AMS to notify all nodes in the message space that the node has rescinded its willingness to accept private messages on the indicated subject from nodes in the domain of the service request.

3.1.6.5 Additional Comments

None.

3.1.7 ASSERT_SUBSCRIPTION.REQUEST

3.1.7.1 Function

The **Assert_subscription.request** primitive shall be used by the node to subscribe to published messages on a specific subject.

3.1.7.2 Semantics

Assert_subscription.request shall provide parameters as follows:

Assert_subscription.request (subject ID,
service mode,
priority,
flow label,
continuum ID of publisher(s),
unit ID of publisher(s),
role ID of publisher(s))

3.1.7.3 When Generated

Assert_subscription.request is generated by the node at any time while the node is participating in its message space.

3.1.7.4 Effect on Reception

Reception of **Assert_subscription.request** shall, if approved, cause AMS to notify all nodes in the message space of the node's subscription to messages on the indicated message subject from nodes in the domain of the service request.

3.1.7.5 Additional Comments

The priority and flow label specified in the **Assert_subscription.request** are advisory, expressing the node's preferences for delivery of messages on this subject.

3.1.8 CANCEL_SUBSCRIPTION.REQUEST

3.1.8.1 Function

The **Cancel_subscription.request** primitive shall be used by the node to terminate its subscription to published messages on a specific subject.

3.1.8.2 Semantics

Cancel_subscription.request shall provide parameters as follows:

Cancel_subscription.request (subject ID,
continuum ID of publisher(s),
unit ID of publisher(s),
role ID of publisher(s))

3.1.8.3 When Generated

Cancel_subscription.request is generated by the node at any time while the node is participating in its message space.

3.1.8.4 Effect on Reception

Reception of **Cancel_subscription.request** shall cause AMS to notify all nodes in the message space that the node is has rescinded its subscription to messages on the indicated message subject from nodes in the domain of the service request

3.1.8.5 Additional Comments

None.

3.1.9 PUBLISH.REQUEST

3.1.9.1 Function

The **Publish.request** primitive shall be used by the node to publish a message.

3.1.9.2 Semantics

Publish.request shall provide parameters as follows:

```
Publish.request (subject ID,
                 [priority],
                 [flow label],
                 content length,
                 [content],
                 [context])
```

3.1.9.3 When Generated

Publish.request is generated by the node at any time while the node is participating in its message space.

3.1.9.4 Effect on Reception

Reception of **Publish.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the content as necessary) and send one copy of that message to every node in the message space that currently subscribes to messages on the indicated message subject from the operative node.

3.1.9.5 Additional Comments

Context, if specified, identifies context information that is meaningful to the publishing node. Priority and flow label, where specified, override the priority and flow label preferences expressed in the applicable subscriptions as asserted by the implied destination nodes.

3.1.10 SEND.REQUEST

3.1.10.1 Function

The **Send.request** primitive shall be used by the node to send a message privately to a single node.

3.1.10.2 Semantics

Send.request shall provide parameters as follows:

Send.request (continuum ID of destination,
unit ID of destination,
node number of destination,
subject ID,
[priority],
[flow label],
content length,
[content],
[context])

3.1.10.3 When Generated

Send.request is generated by the node at any time while the node is participating in its message space.

3.1.10.4 Effect on Reception

Reception of **Send.request** shall, if approved and if the destination node currently invites messages on the indicated subject from the operative node, cause AMS to construct a message as indicated (marshaling the content as necessary) and send it to the specified destination node.

3.1.10.5 Additional Comments

Context, if specified, identifies context information that is meaningful to the sending node. Priority and flow label, where specified, override the priority and flow label preferences expressed in the applicable invitation as asserted by the destination node.

3.1.11 QUERY.REQUEST

3.1.11.1 Function

The **Query.request** primitive shall be used by the node to send a message privately to a single node in a synchronous fashion.

3.1.11.2 Semantics

Query.request shall provide parameters as follows:

```
Query.request (continuum ID of destination,
               unit ID of destination,
               node number of destination,
               subject ID,
               [priority],
               [flow label],
               content length,
               [content],
               [term],
               context)
```

3.1.11.3 When Generated

Query.request is generated by the node at any time while the node is participating in its message space.

3.1.11.4 Effect on Reception

Reception of **Query.request** shall, if approved and if the destination node currently invites messages on the indicated subject from the operative node, cause AMS to construct a message as indicated (marshaling the content as necessary), send it to the specified destination node, and suspend the operation of the querying node until either a reply to this message is received, a notice of delivery failure for this message is received, or – if *term* is specified – the indicated term has expired.

3.1.11.5 Additional Comments

Context identifies context information that is meaningful to the querying node. If term is omitted, **only** reception of a reply or of a notice of delivery failure will cause the operation of the querying node to be resumed. Priority and flow label, where specified, override the priority and flow label preferences expressed in the applicable invitation as asserted by the destination node.

3.1.12 REPLY.REQUEST

3.1.12.1 Function

The **Reply.request** primitive shall be used by the node to reply to a message sent by some node.

3.1.12.2 Semantics

Reply.request shall provide parameters as follows:

```
Reply.request (continuum ID of destination,
               unit ID of destination,
               node number of destination,
               subject ID,
               [priority],
               [flow label],
               content length,
               [content],
               context)
```

3.1.12.3 When Generated

Reply.request is generated by the node at any time while the node is participating in its message space.

3.1.12.4 Effect on Reception

Reception of **Reply.request** shall, if approved and if the destination node currently invites messages on the indicated subject from the operative node, cause AMS to construct a message as indicated (marshaling the content as necessary) and send it to the specified destination node.

3.1.12.5 Additional Comments

Continuum ID, unit ID, and node number must identify the node that sent some previously received message, and context must be the context provided with that message (which will be meaningful only to that node). Priority and flow label, where specified, override the priority and flow label preferences expressed in the applicable invitation as asserted by the destination node.

3.1.13 ANNOUNCE.REQUEST

3.1.13.1 Function

The **Announce.request** primitive shall be used by the node to send a message privately to a set of nodes.

3.1.13.2 Semantics

Announce.request shall provide parameters as follows:

Announce.request (continuum ID,
unit ID,
role ID,
subject ID,
[priority],
[flow label],
content length,
[content],
context)

3.1.13.3 When Generated

Announce.request is generated by the node at any time while the node is participating in its message space.

3.1.13.4 Effect on Reception

Reception of **Announce.request** shall, if approved, cause AMS to construct a message as indicated (marshaling the content as necessary) and send it to every node in the domain of the service request that currently invites messages on the indicated subject from the operative node.

3.1.13.5 Additional Comments

Priority and flow label, where specified, override the priority and flow label preferences expressed in the applicable invitations as asserted by the implied destination nodes.

3.1.14 MESSAGE.INDICATION

3.1.14.1 Function

The **Message.indication** primitive shall be used to deliver AMS original (non-reply) message information to the node.

3.1.14.2 Semantics

Message.indication shall provide parameters as follows:

Message.indication (continuum ID of source,
unit ID of source,
node number of source,
subject ID,
content length,
[content],
[context])

3.1.14.3 When Generated

Message.indication is generated upon reception of an original (non-reply) message from a node.

3.1.14.4 Effect on Reception

The effect on reception of **Message.indication** by a node is undefined.

3.1.14.5 Additional Comments

None.

3.1.15 REPLY.INDICATION

3.1.15.1 Function

The **Reply.indication** primitive shall be used to deliver AMS reply message information to the node.

3.1.15.2 Semantics

Reply.indication shall provide parameters as follows:

Reply.indication (continuum ID of source,
unit ID of source,
node number of source,
subject ID,
content length,
[content],
context)

3.1.15.3 When Generated

Reply.indication is generated upon reception of a reply message from a node.

3.1.15.4 Effect on Reception

The effect on reception of **Reply.indication** by a node is undefined.

3.1.15.5 Additional Comments

Context is the context in which the antecedent message (the message to which the reply message is a response) was sent.

3.1.16 FAULT.INDICATION

3.1.16.1 Function

The **Fault.indication** primitive shall be used to indicate an AMS fault condition to the node.

3.1.16.2 Semantics

Fault.indication shall provide parameters as follows:

Fault.indication (fault expression)

3.1.16.3 When Generated

Fault.indication is generated when AMS encounters a fault condition.

3.1.16.4 Effect on Reception

The effect on reception of **Fault.indication** by a node is undefined.

3.1.16.5 Additional Comments

None.

3.1.17 REGISTER.INDICATION

3.1.17.1 Function

The **Register.indication** primitive shall be used to notify the node of the insertion of some node (including itself) into the message space.

3.1.17.2 Semantics

Register.indication shall provide parameters as follows:

Register.indication (node number of new node,
unit ID of new node,
role ID of new node)

3.1.17.3 When Generated

Register.indication is always generated upon insertion of the node into its own message space. **Register.indication** may optionally also be generated upon insertion of any other node into the message space.

3.1.17.4 Effect on Reception

The effect on reception of **Register.indication** by a node is undefined.

3.1.17.5 Additional Comments

The node number in the **Register.indication** primitive that notifies the node of its own successful registration is the one that the node must provide in the **Unregister.request** primitive.

3.1.18 UNREGISTER.INDICATION

3.1.18.1 Function

The **Unregister.indication** primitive shall be used to notify the node of the removal of some node (including itself) from the message space.

3.1.18.2 Semantics

Unregister.indication shall provide parameters as follows:

Unregister.indication (node number of removed node,
unit ID of removed node)

3.1.18.3 When Generated

Unregister.indication is optionally generated upon removal of any node from the message space.

3.1.18.4 Effect on Reception

The effect on reception of **Unregister.indication** by a node is undefined.

3.1.18.5 Additional Comments

None.

3.1.19 ASSERT_INVITATION.INDICATION

3.1.19.1 Function

The **Assert_invitation.indication** primitive shall be used to notify the node of a newly asserted message invitation in the message space.

3.1.19.2 Semantics

Assert_invitation.indication shall provide parameters as follows:

> **Assert_invitation.indication** (node number of inviting node,
> unit ID of inviting node,
> subject ID,
> service mode,
> priority,
> flow label,
> unit ID of invitee(s),
> role ID of invitee(s))

3.1.19.3 When Generated

Assert_invitation.indication is optionally generated upon assertion of an invitation by some node in the message space.

3.1.19.4 Effect on Reception

The effect on reception of **Assert_invitation.indication** by a node is undefined.

3.1.19.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the node's understanding of the functional 'connections' between itself and other nodes.

3.1.20 CANCEL_INVITATION.INDICATION

3.1.20.1 Function

The **Cancel_invitation.indication** primitive shall be used to notify the node of a newly canceled message invitation in the message space.

3.1.20.2 Semantics

Cancel_invitation.indication shall provide parameters as follows:

Cancel_invitation.indication (node number of disinviting node,
unit ID of disinviting node,
subject ID,
unit ID of invitee(s),
role ID of invitee(s))

3.1.20.3 When Generated

Cancel_invitation.indication is optionally generated upon cancellation of a message invitation by some node in the message space.

3.1.20.4 Effect on Reception

The effect on reception of **Cancel_invitation.indication** by a node is undefined.

3.1.20.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the node's understanding of the functional 'connections' between itself and other nodes.

3.1.21 ASSERT_SUBSCRIPTION.INDICATION

3.1.21.1 Function

The **Assert_subscription.indication** primitive shall be used to notify the node of a newly asserted subscription in the message space.

3.1.21.2 Semantics

Assert_subscription.indication shall provide parameters as follows:

```
    Assert_subscription.indication (node number of subscribing node,
                                    unit ID of subscribing node,
                                    subject ID,
                                    service mode,
                                    priority,
                                    flow label,
                                    continuum ID of publisher(s),
                                    unit ID of publisher(s),
                                    role ID of publisher(s))
```

3.1.21.3 When Generated

Assert_subscription.indication is optionally generated upon assertion of a subscription by some node in the message space.

3.1.21.4 Effect on Reception

The effect on reception of **Assert_subscription.indication** by a node is undefined.

3.1.21.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the node's understanding of the functional 'connections' between itself and other nodes.

The specification of a subscription has two elements: subject number and *publisher profile*. The publisher profile specifies the continuum, unit, and role characterizing the publishers from which messages are exclusively solicited by this subscription; it implicitly defines the subscription's *publisher set*, the set of all nodes in the venture conforming to the publisher profile. A subscription whose publisher set is a superset of some other subscription's publisher set *subsumes* that other subscription.

3.1.22 CANCEL_SUBSCRIPTION.INDICATION

3.1.22.1 Function

The **Cancel_subscription.indication** primitive shall be used to notify the node of a newly canceled subscription in the message space.

3.1.22.2 Semantics

Cancel_subscription.indication shall provide parameters as follows:

Cancel_subscription.indication (node number of unsubscriber,
unit ID of unsubscriber,
subject ID,
continuum ID of publisher(s),
unit ID of publisher(s),
role ID of publisher(s))

3.1.22.3 When Generated

Cancel_subscription.indication is optionally generated upon cancellation of a subscription by some node in the message space.

3.1.22.4 Effect on Reception

The effect on reception of **Cancel_subscription.indication** by a node is undefined.

3.1.22.5 Additional Comments

This indication is provided to facilitate message space configuration monitoring. The information provided in the indication may be used to help manage the node’s understanding of the functional ‘connections’ between itself and other nodes.

3.2 SERVICES REQUIRED OF THE TRANSPORT SERVICE

The service primitives and parameters required to access the services of the Transport service are:

`TRANSPORT.request`	(TSDU, TS Address)
`TRANSPORT.indication`	(TSDU, TS Address)

NOTES

1 The service assumed of the underlying layer is as simple as possible to allow AMS to be as generally applicable as possible. For the purposes of this specification the service is referred to as the Transport Service and the delimited data unit it transfers is the Transport Service Data Unit (TSDU). Although the Transport service is conceptually a single service, in practice AMS will likely use several different underlying communication systems concurrently.

2 The format and contents of the TS Address parameter depend on the addressing capabilities and conventions of the underlying service. Information in the MIB must enable translation between the identifiers of AMS communicating entities and the corresponding TS addresses. The above list of mandatory Transport service primitives does not imply that the Transport service is prohibited from supporting additional services for the convenience of the AMS implementation.

3 The quality of service provided by the transport service may vary, so long as it can be characterized within AMS in terms of the quality of service parameters discussed in 3.1.2.10 through 3.1.2.12 above.

4 PROTOCOL SPECIFICATION

4.1 GENERAL

All Meta-AMS PDUs (or 'MPDUs') shall be transmitted using the AMS continuum's primary transport service.

Reception of an MPDU which is determined to be inauthentic shall cause that PDU to be discarded immediately and processed no further.

An MPDU shall be deemed inauthentic if the sender of the PDU is an AMS communicating entity for which a public key is present in the MIB and either:

– the PDU does not contain a digital signature (see 5.1.4), or

– when the digital signature is decrypted using the applicable public key, the result is not identical to the text string from which the digital signature was generated.

Digital signatures contained in MPDUs from entities for which no public key is present in the MIB shall be ignored.

4.2 CONFIGURATION PROCEDURES

4.2.1 CONFIGURATION SERVER INITIALIZATION

Upon commencing operations and once per minute thereafter, the configuration server shall send an *I_am_running* MPDU to every 'lower-ranking' network location at which the configuration server is authorized to operate, as indicated in the configuration server's Management Information Base.

NOTE – Since these network locations are listed in descending order of preference, a lower-ranking network location is defined as one that appears at some point in the list after the configuration server's own network location.

On reception of an *I_am_running* MPDU, the configuration server shall cease operations.

4.2.2 CONFIGURATION SERVER LOCATION

When the network location of the continuum's configuration server is unknown, contact with the configuration server shall be attempted as follows at each of the network locations at which the configuration server is authorized to operate, as indicated (in descending order of preference) in the node's Management Information Base, cycling through the list as necessary until either contact succeeds or else AMS activity is terminated:

– An *are_you_active* MPDU shall be sent to the indicated network location.

– On reception of an *are_you_active* MPDU, the configuration server shall return a *config_msg_ack* MPDU.

– On reception of a *config_msg_ack* MPDU in response to an *are_you_active*, contact with the configuration server shall be deemed to have succeeded and the network location of the originating configuration server shall be noted. If no responding *config_msg_ack* MPDU is received within N1 seconds, a **Fault.indication** primitive shall be delivered.

4.2.3 REGISTRAR INITIALIZATION

Upon commencing operations, the registrar shall:

– locate the configuration server;

– send an *announce_registrar* MPDU to the configuration server.

On reception of an *announce_registrar* MPDU, the configuration server shall:

– determine whether or not (a) a registrar for the indicated cell of the indicated message space is already known to be running or (b) the indicated cell is unknown to the configuration server (i.e., not identified in the Management Information Base);

– if so, return a *rejection* MPDU indicating the reason for refusing the announcement;

– otherwise:

  • note the registrar for this cell;

  • return a *config_msg_ack* MPDU;

  • send to every other registrar in the message space a *cell_spec* MPDU indicating the network location of the new registrar;

  • commence an N3-second heartbeat cycle for heartbeat exchange with the registrar. (See discussion of heartbeats below.)

On reception of a *rejection* MPDU the registrar shall cease operations.

On reception of a *config_msg_ack* MPDU, the registrar shall:

– commence an N3-second heartbeat cycle for heartbeat exchange with the configuration server; see discussion of heartbeats below;

– send a *msg_space_query* MPDU to the configuration server.

On reception of a *msg_space_query* MPDU from the registrar of some cell, for every other cell in the message space the configuration server shall return one *cell_spec* MPDU indicating the network location of that cell’s registrar.

On reception of a *cell_spec* MPDU, the registrar shall note the network location of the indicated cell’s registrar.

4.2.4 REGISTRAR LOCATION

When the network location of a cell's registrar is unknown:

- The network location of the configuration server shall be determined as necessary.
- A *registrar_query* MPDU shall be sent to the configuration server.
- On reception of a *registrar_query* MPDU, the configuration server shall return a *cell_spec* MPDU indicating the network location of the registrar.
- On reception of a *cell_spec* MPDU in response to a *registrar_query*, the network location of the registrar shall be noted. If no responding *cell_spec* MPDU is received within N1 seconds, a **Fault.indication** primitive shall be delivered.
- This procedure shall be repeated until successful (i.e., no **Fault.indication** primitive is delivered) or until AMS activity is terminated.

4.2.5 NODE REGISTRATION

On reception of a **Register.request** primitive:

- If the role ID indicates 'all roles' or is otherwise invalid, a **Fault.indication** primitive shall be delivered and the request shall be processed no further.
- The network location of the registrar for the node's cell shall be determined.
- The node shall send a *node_registration* MPDU to that registrar.
- On reception of a *node_registration* MPDU:
  - If the number of nodes in the cell's registered membership is already equal to the maximum permitted (an implementation matter), the registrar shall return a *rejection* MPDU indicating the reason for refusing the registration.
  - If the registrar has been operating for less than N5 seconds (three times the node heartbeat period) then it may not yet have an authoritative list of all nodes in the cell's registered membership and therefore cannot yet assign node numbers without risk of duplication. In this case the registrar shall return a *rejection* MPDU indicating the reason for refusing the registration.
  - Otherwise:
    - The registrar shall assign an unused node number to the node and return a *you_are_in* MPDU, indicating the node's own node number. At this point the node has become a member of the registered membership of the registrar's cell.
    - The registrar shall commence an N4-second heartbeat cycle for heartbeat exchange with the new node; see discussion of heartbeats below.

- On reception of a *rejection* MPDU a **Fault.indication** primitive shall be delivered and the registration attempt may be either abandoned or repeated (an implementation decision).
- On reception of a *you_are_in* MPDU:
  - The node shall commence an N4-second heartbeat cycle for heartbeat exchange with the registrar; see discussion of heartbeats below.
  - An *I_am_starting* MPDU containing the node's MAMS endpoint name shall be sent to the registrar. This MPDU shall be re-sent to the registrar every N2 seconds until a responding *config_msg_ack* MPDU is received.
  - An invitation for messages on subject zero, from nodes characterized by role '1' (the RAMS gateway) in the root cell, shall be asserted as described in 3.1.5 above. These messages will contain 'enclosures' that are messages issued in remote continua and forwarded via RAMS procedures.
  - At this time the node may commence application-specific message subscription, invitation, transmission, and reception activity.
- On a registrar's reception of an *I_am_starting* MPDU:
  - If the MPDU's originating node is a member of the registrar's cell's registered membership, then the registrar shall return a *config_msg_ack* MPDU to that originating node and shall forward the *I_am_starting* MPDU to every other node in its cell's registered membership and to every other registrar in the message space.
  - Otherwise, the registrar shall return a *you_are_dead* MPDU to the MPDU's originating node.
- On reception of an *I_am_starting* MPDU from another registrar, the registrar shall forward the MPDU to every node in its cell's registered membership.
- On a node's reception of an *I_am_starting* MPDU from a registrar:
  - The new node shall be noted.
  - Optionally, a **Register.indication** primitive shall be delivered.
  - An *I_am_here* MPDU containing the receiving node's MAMS endpoint name, all of its non-empty delivery vectors, and all of its current invitations and subscriptions shall be sent to the new node.
- On a node's reception of a *I_am_here* MPDU:
  - The issuing node's MAMS endpoint name shall be noted.
  - Optionally, a **Register.indication** primitive shall be delivered.

  - For each of the originating node's delivery vectors (i.e., supported service modes), the most preferred delivery point to which the receiving node is able to send messages – the 'best fit' delivery point – shall be noted. The 'best fit' delivery point is determined by examining the delivery points in the delivery vector, in the order in which they appear, until one is encountered whose transport service name is equal to that of one of the transport services by which the receiving node can transmit AMS messages, as indicated in the node's Management Information Base. If there is no 'best fit' delivery point for this service mode, then this fact shall be noted.
  - For each of the originating node's subscriptions:
    - The subscription shall be noted.
    - Optionally, an **Assert_subscription.indication** primitive shall be delivered.
  - For each of the originating node's invitations:
    - The invitation shall be noted.
    - Optionally, an **Assert_invitation.indication** primitive shall be delivered.
  - A *declaration* PDU containing all of the receiving node's non-empty delivery vectors, invitations (possibly an empty list at this time), and subscriptions (again, possibly an empty list at this time) shall be returned to the node that sent the *I_am_here* MPDU.

– On reception of a *declaration* MPDU:

  - For each of the originating node's delivery vectors (i.e., supported service modes), the most preferred delivery point to which the receiving node is able to send messages – the 'best fit' delivery point – shall be noted as described above.
  - For each of the originating node's subscriptions:
    - The subscription shall be noted.
    - Optionally, an **Assert_subscription.indication** primitive shall be delivered.
  - For each of the originating node's invitations:
    - The invitation shall be noted.
    - Optionally, an **Assert_invitation.indication** primitive shall be delivered.

4.2.6 UNREGISTRATION

On reception of an **Unregister.request** primitive:

- The network location of the registrar shall be determined.
- An *I_am_stopping* MPDU shall be sent to the registrar.
- The node's AMS activity shall cease immediately.

On reception of an *I_am_stopping* MPDU from a node in its cell's registered membership, the registrar shall note the termination of the node and shall forward the MPDU to every other node in its cell's registered membership and to every other registrar in the message space.

On reception of an *I_am_stopping* MPDU from another registrar, the registrar shall forward the MPDU to every node in its cell's registered membership.

On a node's reception of an *I_am_stopping* MPDU from a registrar:

- The node's removal shall be noted.
- Optionally, an **Unregister.indication** primitive shall be delivered.
- For each of the terminated node's invitations:
  - The invitation shall be forgotten.
  - Optionally, a **Cancel_invitation.indication** primitive shall be delivered.
- For each of the terminated node's subscriptions:
  - The subscription shall be forgotten.
  - Optionally, a **Cancel_subscription.indication** primitive shall be delivered.
- If the receiving node is the terminated node itself (i.e., the node's termination was imputed from a failure to deliver heartbeats to the registrar on a timely basis, as discussed below), the node's AMS activity shall cease immediately.

4.2.7 HEARTBEATS

Upon expiration of any heartbeat period, a *heartbeat* MPDU shall be sent to the relevant communicating entity and a reciprocating *heartbeat* MPDU shall be expected from that entity. Whenever three successive heartbeat periods lapse after reception of a *heartbeat* MPDU before reception of another, an unplanned termination of the relevant communicating entity is imputed.

Imputed termination of the configuration server shall simply cause the network location of the configuration server to be unknown for future communication purposes. (This will eventually force registrars to re-locate the configuration server and re-register with it, and in so doing it may additionally force nodes to re-locate the configuration server in order to re-establish communications with unresponsive registrars.)

Upon imputed termination of a registrar, the configuration server shall take some action intended to cause the registrar to be restarted; the nature of the action to be taken is an implementation matter.

The effect of imputed registrar termination on each node in the registrar's cell shall be as follows:

– The restarted registrar shall be located.

– A *reconnect* MPDU indicating the numbers of all nodes in the cell's registered membership shall be sent to the restarted registrar.

On reception of a *reconnect* MPDU from a node, the registrar shall:

– return a *you_are_dead* MPDU if the registrar has been operating for more than N5 seconds, during which time all nodes in the cell should already have contacted it (this serves to defend the registrar against spurious reconnections);

– return a *you_are_dead* MPDU if a *reconnect* MPDU had previously been received from some other node, and the node census in that *reconnect* MPDU did not include the node that issued this new *reconnect* MPDU (this serves to defend the registrar against inconsistent cell censuses);

– otherwise:

  • commence an N4-second heartbeat cycle for heartbeat exchange with this node;

  • return a *config_msg_ack* MPDU.

On reception of a *config_msg_ack* MPDU in response to a *reconnect* MPDU, the receiving node shall re-commence its N4-second heartbeat cycle for heartbeat exchange with the registrar.

Upon imputed termination of a node, the registrar shall send an *I_am_stopping* MPDU on behalf of this node to every other node in its cell and to every other registrar in the message space. This will have the effect of unregistering the node in the manner discussed above.

On reception of a *heartbeat* MPDU from an unregistered registrar, the configuration server shall return a *you_are_dead* MPDU.

On reception of a *heartbeat* MPDU from an unregistered node, the receiving registrar shall return a *you_are_dead* MPDU.

On reception of a *you_are_dead* MPDU, the receiving communicating entity shall cease AMS activity immediately.

4.2.8 RESYNCHRONIZATION

Each registrar, upon expiration of its configuration resynchronization interval (as indicated in the registrar's Management Information Base), shall:

– Send a *cell_status* MPDU to every node in its cell's registered membership and to every other registrar in the message space.

On reception of a *cell_status* MPDU from another registrar, the receiving registrar shall forward the MPDU to every node in its cell's registered membership.

On reception of a *cell_status* MPDU at a node, the receiving node shall:

– for each node in the referenced cell whose registration is currently noted but whose node number does not appear in the MPDU's node list, simulate reception of an *I_am_stopping* MPDU from this node as described in 4.2.6 above;

– if the referenced cell is the node's own cell, return a node_status MPDU to that cell's registrar.

On reception of a *node_status* MPDU from a node in its own cell's registered membership, the registrar shall forward the MPDU to every other node in its cell's registered membership and to every other registrar in the message space.

On reception of a *node_status* MPDU from another registrar, the receiving registrar shall forward the MPDU to every node in its cell's registered membership.

On reception of a *node_status* MPDU at a node, the receiving node shall:

– if the declared MAMS endpoint name and/or role of the MPDU conflict with those currently noted for the indicated node, simulate reception of an *I_am_stopping* MPDU from this node as described in 4.2.6 above;

– if, at this point, the indicated node is unknown, note the node;

– for each currently noted subscription for the indicated node that does not appear in the MPDU's subscription list, simulate reception of an *unsubscribe* MPDU for this subscription for this node as described in 4.2.10 below;

– for each currently noted invitation for the indicated node that does not appear in the MPDU's invitation list, simulate reception of a *disinvite* MPDU for this invitation for this node as described in 4.2.12 below;

– for each subscription in the MPDU's subscription list that is not currently noted by the operative node, simulate reception of a *subscribe* MPDU for this subscription for this node as described in 4.2.9 below;

– for each invitation in the MPDU's invitation list that is not currently noted by the operative node, simulate reception of an *invite* MPDU for this invitation for this node as described in 4.2.11 below.

4.2.9 SUBSCRIPTION ASSERTION

On reception of an **Assert_subscription.request** primitive:

– If the operative node is not an authorized receiver of messages on this subject, or if the subject ID of the request indicates 'all subjects' and the continuum ID of the request indicates either 'all continua' or any continuum other than the local continuum, then a **Fault.indication** primitive shall be delivered.

– Otherwise:

  • The network location of the registrar shall be determined.

  • A *subscribe* MPDU shall be sent to the registrar.

  • Optionally, an **Assert_subscription.indication** primitive shall be delivered, notifying the operative node of its own subscription assertion.

NOTE – A node may issue subscriptions whose domains are not disjoint sets. Selection of delivery vector, priority, and flow label when sending a message that satisfies multiple, possibly conflicting subscriptions from the same subscriber is an implementation matter.

On a registrar's reception of a *subscribe* MPDU from a node in its cell's registered membership, the registrar shall forward the message to every other node in its cell's registered membership and to every other registrar in the message space.

On reception of a *subscribe* MPDU from another registrar, the registrar shall forward the message to every node in its cell's registered membership.

On a node's reception of a *subscribe* MPDU from a registrar:

– The subscription shall be noted. A subscription to the subject whose number is zero shall signify a subscription to messages on all subjects.

– Optionally, an **Assert_subscription.indication** primitive shall be delivered.

4.2.10 SUBSCRIPTION CANCELLATION

On reception of a **Cancel_subscription.request** primitive:

– If no subscription for messages on the indicated subject is currently asserted – that is, asserted and not subsequently canceled – for the indicated continuum, unit, and role, then the cancellation shall be judged redundant and a **Fault.indication** primitive shall be delivered.

– Otherwise:

  • The network location of the registrar shall be determined.

  • An *unsubscribe* MPDU shall be sent to the registrar.

    - Optionally, a **Cancel_subscription.indication** primitive shall be delivered, notifying the operative node of its own subscription cancellation.

On reception of an *unsubscribe* MPDU from a node in its cell's registered membership, the registrar shall forward the message to every other node in its cell's registered membership and to every other registrar in the message space.

On reception of an *unsubscribe* MPDU from another registrar, the registrar shall forward the message to every node in its cell's registered membership.

On a node's reception of an *unsubscribe* MPDU from a registrar:

– The indicated subscription shall be forgotten.

– Optionally, a **Cancel_subscription.indication** primitive shall be delivered.

4.2.11 INVITATION ASSERTION

On reception of an **Assert_invitation.request** primitive:

– If the operative node is not an authorized receiver of messages on this subject, a **Fault.indication** primitive shall be delivered.

– Otherwise:

    - The network location of the registrar shall be determined.
    - An *invite* MPDU shall be sent to the registrar.
    - Optionally, an **Assert_invitation.indication** primitive shall be delivered, notifying the operative node of its own invitation assertion.

NOTE – A node may issue invitations whose domains are not disjoint sets. Selection of delivery vector, priority, and flow label when sending a message that satisfies multiple, possibly conflicting invitations from the same inviter is an implementation matter.

On a registrar's reception of an *invite* MPDU from a node in its cell's registered membership, the registrar shall forward the message to every other node in its cell's registered membership and to every other registrar in the message space.

On reception of an *invite* MPDU from another registrar, the registrar shall forward the message to every node in its cell's registered membership.

On a node's reception of an *invite* MPDU from a registrar:

– The invitation shall be noted.

– Optionally, an **Assert_invitation.indication** primitive shall be delivered.

4.2.12 INVITATION CANCELLATION

On reception of a **Cancel_invitation.request** primitive:

– If no invitation for messages on the indicated subject is currently asserted – that is, asserted and not subsequently canceled – for the indicated unit and role, then the cancellation shall be judged redundant and a **Fault.indication** primitive shall be delivered.

– Otherwise:

  • The network location of the registrar shall be determined.

  • A *disinvite* MPDU shall be sent to the registrar.

  • Optionally, a **Cancel_invitation.indication** primitive shall be delivered, notifying the operative node of its own invitation cancellation.

On reception of a *disinvite* MPDU from a node in its cell's registered membership, the registrar shall forward the message to every other node in its cell and to every other registrar in the message space.

On reception of a *disinvite* MPDU from another registrar, the registrar shall forward the message to every node in its cell's registered membership.

On a node's reception of a *disinvite* MPDU from a registrar:

– The indicated invitation shall be forgotten.

– Optionally, a **Cancel_invitation.indication** primitive shall be delivered.

4.3 COMMUNICATION PROCEDURES

4.3.1 MESSAGE TRANSMISSION

Whenever an AMS message is to be transmitted from one node to another, the transport service to use for this transmission shall be selected as follows:

– If the delivery vector (i.e., service mode) specified for this message transmission activity by the destination node is one for which no 'best fit' delivery point could be determined, then transmission of the message is impossible; a **Fault.indication** primitive shall be delivered.

– Otherwise, the 'best fit' delivery point shall be used as the destination endpoint for the message and the indicated transport service shall be used to accomplish the transmission. If the transmission is determined to have failed for any reason, a **Fault.indication** primitive shall be delivered.

NOTE – A node may issue subscriptions and invitations whose domains are not disjoint sets. Selection of delivery vector, priority, and flow label when sending a message that satisfies multiple, possibly conflicting subscriptions and/or invitations from the same node is an implementation matter.

Authorization to send or receive messages on a given subject shall be determined by reference to security information in the MIB where present.

Direction to marshal and un-marshal messages on a given subject, and the manner in which such marshaling and un-marshaling is to be accomplished, shall be determined by reference to subject catalog information in the MIB where present.

Direction to encrypt and decrypt messages on a given subject, and the manner in which such encryption and decryption is to be accomplished, shall be determined by reference to security information in the MIB where present.

4.3.2 PUBLISH

On reception of a **Publish.request** primitive:

– If the node is not an authorized sender of messages on this subject, a **Fault.indication** primitive shall be delivered.

– Otherwise, an AMS message shall be constructed using the provided parameters, with content marshaled and/or encrypted as necessary, and one copy of this message shall be transmitted to every node in the message space that is an authorized receiver of messages on the indicated subject for which the operative node has noted at least one currently asserted subscription (a) that is for messages on this subject or on all subjects and (b) whose domain includes the operative node.

4.3.3 REMOTE AMS PRIVATE MESSAGE TRANSMISSION

Whenever a service primitive is received that requests that a message be sent privately (as described in subsections 4.3.4, 4.3.5, 4.3.6, and 4.3.7 below) to a node or nodes residing in some continuum other than the local continuum:

– An *enclosure* data structure shall be constructed from the parameters of the service primitive, in exactly the same way that an AMS message would be constructed if the destination node resided in the local continuum.

– An *envelope* data structure shall be constructed whose content is the newly constructed enclosure; see 5.3.2 below. The destination continuum number, unit number, and node number or role number of the envelope shall be those identifying the destination of the enclosure; the subject number of the envelope shall identify the subject specified in the original service primitive. The control code of the envelope shall be '5' in the event that the original service primitive specified destination node number (send, query, or reply); it shall be '6' in the event that the original service primitive specified destination role number (announce).

– A message shall be published whose content is the envelope constructed above. If the destination continuum number is zero, signifying 'all continua', then the subject of the published message shall be the additive inverse of the number of the local continuum. Otherwise, the subject of the published message shall be the additive inverse of the destination continuum number.

4.3.4 SEND

On reception of a **Send.request** primitive:

– If the node is not an authorized sender of messages on this subject or the indicated destination node is not an authorized receiver of messages on this subject, a **Fault.indication** primitive shall be delivered.

– Otherwise, if the indicated destination continuum is not the local continuum, then the remote AMS private message transmission procedure described in 4.3.3 above shall be performed.

– Otherwise, if the operative node has noted no current invitation from the indicated destination node (a) that is for messages on the indicated subject and (b) whose domain includes the operative node, a **Fault.indication** primitive shall be delivered.

– Otherwise, an AMS message shall be constructed using the provided parameters, with content marshaled and/or encrypted as necessary, and the message shall be transmitted to the indicated destination node.

4.3.5 QUERY

On reception of a **Query.request** primitive:

– If the node is not an authorized sender of messages on this subject or the indicated destination node is not an authorized receiver of messages on this subject, or if context number is zero or is equal to the context associated with some other pending query for this node, or if the indicated destination continuum is the local continuum and the operative node has noted no current invitation from the indicated destination node (a) that is for messages on the indicated subject and (b) whose domain includes the operative node, a **Fault.indication** primitive shall be delivered.

– Otherwise:

  • If the indicated destination continuum is not the local continuum, then the remote AMS private message transmission procedure described in 4.3.3 above shall be performed; otherwise an AMS message shall be constructed using the provided parameters, with content marshaled and/or encrypted as necessary, and the message shall be transmitted to the indicated destination node.

  - Handling of inbound AMS messages by the operative node shall be suspended pending arrival of a reply message in response to this query message.
  - If a term was specified in the **Query.request** primitive, a query countdown timer for the indicated interval shall be started.

On expiration of a query timer:

- A **Fault.indication** primitive shall be delivered indicating the timeout.
- The timer shall be destroyed.
- Handling of inbound AMS messages by the operative node shall be resumed.

4.3.6 REPLY

On reception of a **Reply.request** primitive:

- If the node is not an authorized sender of messages on this subject or the indicated destination node is not an authorized receiver of messages on this subject, or if the value of the context parameter is zero, a **Fault.indication** primitive shall be delivered.
- Otherwise, if the indicated destination continuum is not the local continuum, then the remote AMS private message transmission procedure described in 4.3.3 above shall be performed.
- Otherwise, if the operative node has noted no current invitation from the indicated destination node (a) that is for messages on the indicated subject and (b) whose domain includes the operative node, a **Fault.indication** primitive shall be delivered.
- Otherwise, an AMS message shall be constructed using the provided parameters, with content marshaled and/or encrypted as necessary, and the message shall be transmitted to the indicated destination node.

4.3.7 ANNOUNCE

On reception of an **Announce.request** primitive:

- If the node is not an authorized sender of messages on this subject, a **Fault.indication** primitive shall be delivered.
- Otherwise:
  - If the indicated continuum is either the local continuum or ‘all continua’ then:
    - An AMS message shall be constructed using the provided parameters, with content marshaled and/or encrypted as necessary.

    ▪ One copy of this message shall be transmitted to every node in the message space that is in the domain of the service request, that is an authorized receiver of messages on the indicated subject, and that is a node for which the operative node has noted at least one current invitation (a) that is for messages on the indicated subject and (b) whose domain includes the operative node.

  • If the indicated continuum is either 'all continua' or any continuum other than the local continuum, then the remote AMS private message transmission procedure described in 4.3.3 above shall also be performed.

4.3.8 RECEIVE

On reception of an AMS message:

– If the indicated source node is not an authorized sender of messages on this subject a **Fault.indication** primitive shall be delivered. The message shall be discarded without further processing.

– If and only if the sender of the message is a RAMS gateway (that is, a node whose role number is '1'), indicating that the content of the message is an enclosure, then that enclosure shall be the message to which the rest of this subsection pertains. In effect, the 'outer' header of the message is stripped off and discarded before processing continues.

– The message's content shall be decrypted and/or unmarshaled as necessary.

– If the message's 'reply' flag is '1':

  • If handling of inbound AMS messages by the operative node is suspended because of a query request, and the received message's context number is equal to the context number of the query:

    ▪ Handling of inbound AMS messages by the operative node shall be resumed.

    ▪ The current query timer, if any, shall be stopped and destroyed.

  • A **Reply.indication** primitive shall be delivered.

– Otherwise, a **Message.indication** primitive shall be delivered.

4.4 REMOTE AMS GATEWAY PROCEDURES

4.4.1 OVERVIEW

As noted earlier, RAMS procedures are executed by special-purpose application nodes called *RAMS gateways*.

No single message space may contain more than one RAMS gateway. RAMS gateway behavior is undefined when two or more RAMS gateways are registered within any single message space.

The protocol data units used to effect interoperation among RAMS gateways are termed RAMS PDUs, or *RPDUs*. RPDUs shall be exchanged among RAMS gateways by means of RAMS communication channels (or, in this document, simply *channels*). A channel is the configuration of the communications hardware and software at two RAMS gateways such that RPDUs may be reliably conveyed from one gateway to the other without being relayed by any other RAMS gateway.

The set of RAMS gateways for all of the message spaces for some venture and no message space of any other venture is termed the *RAMS network* for that venture. Each member of a RAMS network must have a channel to at least one other member of the set, and no member of a RAMS network may have channels to any RAMS gateways that are not members of the set. Every member of the RAMS network to which a given RAMS gateway has a channel is termed a *neighbor* of that gateway.

Each RAMS gateway has interfaces to two communication environments: its RAMS network and the AMS message space it serves. These interfaces enable the RAMS gateway to receive AMS messages produced by the nodes in its message space and forward them to nodes in other message spaces of the same venture – directly or indirectly – by encapsulating those messages in RPDUs and sending the RPDUs to its neighbors.

The RAMS network for a given venture must be configured either as a mesh (all RAMS gateways in the network are neighbors of one another) or as a tree (some pairs of RAMS gateways can only intercommunicate through one or more other forwarding gateways, but there is always exactly one 'route' from any gateway in the venture to any other).

As explained below, configuring a RAMS network as a mesh assures that no RPDU will ever traverse more than one channel, but (absent an underlying multicast system) the transmission burden placed on each gateway in the network increases as the number of interconnected message spaces increases. Configuring a RAMS network as a tree enables the transmission burden placed on each gateway in the network to be kept small no matter how many message spaces are included in the venture, so the venture can scale up indefinitely; however, introducing the possibility that messages may traverse multiple channels increases the venture's maximum possible message delivery latency.

In concept, the state of a RAMS gateway is a set of *petitions*. A petition for a given subscription specification (see 3.1.21.5 above) is a summary of the inter-continuum interest, as known to this RAMS gateway, in all AMS messages satisfying that subscription specification. A petition 'for' a given subject is a petition for a subscription specification that cites that subject.

Each petition has three components:

– *SGS* (source gateway set), a list of all neighbors to which the local gateway has asserted (and not yet canceled) this petition.

– *DGS* (destination gateway set), a list of all neighbors that have asserted (and not yet canceled) this petition.

– *DNS* (destination node set), a list of all AMS nodes in the local continuum that have asserted (and not yet canceled) subscriptions that (a) are characterized by this petition's subscription specification and (b) cite in their domains either 'all continua' or some continuum other than the local continuum.

A petition is considered to be in *asserted* state while (and only while) its SGS is non-empty. If the RAMS network is configured as a mesh, the petition is *assertable* while (and only while) its DNS is non-empty; if the RAMS network is configured as a tree, the petition is *assertable* while (and only while) either its DGS or its DNS is non-empty.

The *pseudo-subject* for a continuum is the AMS message subject whose number is the additive inverse of the number of that continuum.

4.4.2 DECLARATION AND RETRACTION

4.4.2.1 Initial Declaration

Upon instantiation, the RAMS gateway for a message space identified within its continuum by a given application name and authority name shall obtain from its Management Information Base (MIB) the RAMS network locations of all currently known neighbors.

The new RAMS gateway shall subscribe to messages on the pseudo-subject for the local continuum, from all nodes in all message spaces of the venture. (Messages published locally on this subject will contain announcements destined for nodes in all continua.)

The RAMS gateway shall add itself to the DNS of the *declaration petition*. The declaration petition is a petition for the subscription specification whose subject is the pseudo-subject for the gateway's local continuum and whose domain is all nodes in all continua.

The RAMS gateway shall *declare* itself to each of its currently known neighbors. A RAMS gateway shall declare itself to a neighbor by sending that neighbor an assertion (see 5.3.3) of the declaration petition and adding that neighbor to the SGS of that petition.

4.4.2.2 Subsequent Declaration Activity

Upon insertion of a new neighbor into the MIB, the RAMS gateway shall declare itself to that neighbor.

Upon removal of a neighbor from the MIB, the RAMS gateway shall *retract* itself from that neighbor. A RAMS gateway shall retract itself from a neighbor by sending that neighbor a cancellation (see 5.3.3) of the declaration petition and removing that neighbor from the SGS of that petition.

Upon termination, the RAMS gateway shall remove itself from the DNS of the declaration petition and retract itself from each of its neighbors.

4.4.2.3 Derived Definitions

A *declared* neighbor is a neighbor whose declaration has been asserted and not yet canceled.

The *conduit* for a continuum is the sole member of the DGS of the petition for the pseudo-subject for that continuum[3].

4.4.3 PETITION ASSERTION AND CANCELLATION

4.4.3.1 Computing the Assertion Set

The Assertion Set (AS) for a petition shall be computed as follows:

– The AS is initially the empty set.

– If the continuum cited in the domain of the petition's subscription specification is 'all continua', add all declared neighbors to the AS. Otherwise add to the AS only the conduit for the continuum cited in the domain.

– Remove from the AS all members of the SGS of this petition.[4]

– If the petition's DNS is empty and its DGS has exactly one member, remove from the AS the sole member of the DGS[5].

[3] I.e., it is the declared neighbor to which RAMS PDUs destined (ultimately) for that continuum are sent.
[4] These gateways are already forwarding all messages that satisfy the subscription specification of the petition; there's no need to assert to them again.
[5] If the only destination for these messages is this single gateway, then we don't need that gateway to send these messages to us (only to be sent back again).

4.4.3.2 Asserting a Petition

When a petition is to be asserted, the following procedure shall be performed:

– The AS for this petition shall be computed as described above.

– An assertion of the petition shall be sent to all members of the computed AS[6].

– The SGS of the petition shall be changed to the union of itself and the computed AS.

4.4.3.3 Computing the Cancellation Set

The Cancellation Set (CS) for a petition shall be computed as follows:

– If the petition's DGS is empty, then the CS shall contain all members of the petition's SGS.

– If, instead, the petition's DGS contains only a single neighbor and that neighbor is also a member of the petition's SGS, then the CS shall contain only the sole member of the DGS[7].

– Otherwise, the CS shall be the empty set.

4.4.3.4 Canceling a Petition

When a petition is to be cancelled, the following procedure shall be performed:

– The CS for this petition shall be computed as described above.

– A cancellation of the petition shall be sent to all members of the computed CS.

– All members of the computed CS shall be removed from the SGS of the petition.

4.4.4 ASSERTION REPORTING

Upon delivery of an **Assert_subscription.indication** primitive to the RAMS gateway that is **not** the result of one of the RAMS gateway's own subscription assertions, where the continuum cited in the domain of the subscription specification is not the local continuum:

– The subscribing node shall be added to the DNS of the petition for this subscription specification.

– The petition shall be asserted as described above[8].

[6] Note that the computed AS may be the empty set, in which case no RPDUs are sent.
[7] Again, if the only destination for these messages is this single gateway, then we don't need that gateway to send these messages to us (only to be sent back again).
[8] Note that the petition is known to be assertable at this time, regardless of the configuration of the RAMS network, because its DNS is now non-empty.

4.4.5 CANCELLATION REPORTING

Upon delivery of a **Cancel_subscription.indication** primitive to the RAMS gateway that is **not** the result of one of the RAMS gateway's own subscription cancellations, where the continuum cited in the domain of the subscription specification is not the local continuum:

– The canceling node shall be removed from the DNS of the petition for this subscription specification.

– If the petition's DNS is now empty then the petition shall be canceled as described above.

4.4.6 ASSERTION REPLICATION

Upon reception of a petition assertion RPDU from a neighbor:

– The neighbor shall be added to the DGS of this petition.

– If the DGS of this petition has one member (i.e., it was formerly empty but is now non-empty) and the domain continuum number cited in the petition's subscription specification is either the local continuum number or zero (indicating 'all continua'), the RAMS gateway shall submit a Subscribe request whose parameters are obtained from the subscription specification for this petition.

– If the subject cited in the subscription specification for this petition is the pseudo-subject for some continuum – i.e., the neighbor is declaring itself to be the conduit for that continuum[9] – then all relevant petition relationships with this neighbor must be established. For each assertable petition:

  • The AS for this assertable petition shall be computed as described above.

  • If the neighbor is a member of the computed AS, then an assertion of this assertable petition shall be sent to the neighbor and the neighbor shall be added to the SGS of this assertable petition.

– Finally, if the petition asserted by the neighbor is itself assertable, then the petition shall be asserted as described above[10].

4.4.7 CANCELLATION REPLICATION

Upon reception of a petition cancellation RPDU from a neighbor:

– The neighbor shall be removed from the DGS of this petition.

[9] Possibly its own local continuum.
[10] This has the effect of propagating the petition assertion throughout the RAMS network if and only if the RAMS network is configured as a tree.

– If the DGS of this petition has zero members (i.e., it was formerly non-empty but is now empty) and the domain continuum number cited in the petition's subscription specification is either the local continuum number or zero (indicating 'all continua'), the RAMS gateway shall submit an Unsubscribe request whose parameters are obtained from the subscription specification for this petition.

– If the subject cited in the subscription specification for this petition is the pseudo-subject for some continuum – i.e., the neighbor is declaring itself no longer to be the conduit for that continuum, which is termed 'the retracting continuum' – then all affected petition relationships with this neighbor must be severed. The set of petitions that are affected by the cancellation shall be computed as follows:

  • If the retracting continuum is the canceling neighbor's own local continuum, then every asserted petition whose SGS contains this neighbor is a member of this set (and no other petition is a member of this set).

  • Otherwise, every asserted petition whose subscription specification's domain continuum number is the number of the retracting continuum **and** whose SGS contains the canceling neighbor is a member of this set (and no other petition is a member of this set).

– For each member of the set of affected petitions:

  • A cancellation of this affected petition shall be sent to the canceling neighbor.

  • The canceling neighbor shall be removed from the SGS of this affected petition.

– Finally, if the DNS of the petition canceled by the neighbor is empty then the petition shall be canceled as described above.

4.4.8 FORWARDING OF PRIVATE AND ANNOUNCED MESSAGES VIA RAMS

On receiving an AMS message on a subject that is the pseudo-subject for some continuum, the RAMS gateway shall proceed as follows:

– The content of the received message must be an envelope data structure with control code indicating either 'send on reception' or 'announce on reception', whose content is an enclosure data structure constituting a message that is intended for private delivery or announcement rather than re-publication. (See 5.3.3 below.)

– If the source node of the received message is not an authorized sender of messages on the subject of the envelope data structure, then the received message shall be discarded.

– Otherwise:

  • If the continuum number in the envelope header is zero, indicating announcement to all continua, then the RAMS gateway shall forward the envelope data structure – an RPDU – to all declared neighbors.

    - Otherwise the RAMS gateway shall forward the envelope data structure – an RPDU – to the conduit for the indicated continuum.

4.4.9 FORWARDING OF PUBLISHED MESSAGES VIA RAMS

On receiving an AMS message on a subject that is greater than zero (i.e., a message on some application-defined subject), the RAMS gateway shall proceed as follows:

- An enclosure data structure shall be constructed from the parameters of the received message, exactly re-creating the message as originally transmitted.
- For each neighbor that is a member of the DGS of at least one petition whose subscription specification is satisfied by the header of the received message:
    - The RAMS gateway shall construct an envelope data structure with control code ‘4’ (‘publish on reception’) whose content is the newly constructed enclosure; see 5.3.2 below.
    - The RAMS gateway shall send the newly constructed envelope data structure – an RPDU – to that neighbor.

4.4.10 FORWARDING FROM NEIGHBORS

Upon reception of an RPDU with control code greater than ‘3’ from a neighbor, the RAMS gateway shall proceed as follows:

- If the control code of the RPDU is ‘4’ (‘publish on reception’), then:
    - The content of the RPDU must be an enclosure data structure constituting a published message from a remote continuum.
    - If the RAMS network is configured as a tree, then for each neighbor – other than the one from which the RPDU was received – that is a member of the DGS of at least one petition whose subscription specification is satisfied by the header of this enclosure (source continuum number, unit number, and role number), the RAMS gateway shall forward the received RPDU to that neighbor.
    - For each node that is a member of the DNS of at least one petition whose subscription specification is satisfied by the header of this enclosure (source continuum number, unit number, and role number), the RAMS gateway shall submit a Send request causing the enclosure to be sent as the content of a message destined for this node; the subject number of this message shall be zero.

NOTE – The enclosure cannot simply be locally re-published by the RAMS gateway: the RAMS gateway itself might not be in the domain of one or more of the subscriptions asserted by members of the applicable destination set, and in any event this re-publication would conceal from the receiving nodes the identity of the original publisher. Using zero as the subject of the private transmission assures that the message can be successfully transmitted to all indicated recipients (see 4.2.5 above, reception of the *you_are_in* MPDU).

– If the control code of the RPDU is '5' ('send on reception'), then the intent is private message delivery rather than re-publication. If the continuum number in the envelope header is not equal to the local continuum number[11], then the RAMS gateway shall forward the received RPDU to the conduit for the indicated continuum. Otherwise, the RAMS gateway shall submit a Send request; the destination unit and node parameters of the request shall be as indicated by the header of the RPDU, the destination continuum shall be the local continuum, the subject shall be zero, and the content to be sent shall be the enclosure data structure that is the content of the RPDU.

– If the control code of the RPDU is '6' ('announce on reception'), then the intent is message announcement rather than re-publication.

  • If the RAMS network is configured as a tree and the continuum number in the envelope header is zero (indicating 'all continua'), then the RAMS gateway shall forward the received RPDU to all of its declared neighbors except the one from which the RPDU was received.

  • If the continuum number in the envelope header is not zero and is also not the number of the local continuum[12], then the RAMS gateway shall forward the received RPDU to the conduit for the indicated continuum. Otherwise (i.e., the continuum number in the envelope header is either zero or the number of the local continuum), the RAMS gateway shall submit an Announce request; the destination unit and role parameters of the request shall be as indicated by the header of the RPDU, the destination continuum shall be the local continuum, the subject shall be zero, and the content to be announced shall be the enclosure data structure that is the content of the RPDU.

[11] Note that this is possible only when the RAMS network is configured as a tree.
[12] Note that this is possible only when the RAMS network is configured as a tree.

5 PROTOCOL DATA UNITS

5.1 META-AMS PROTOCOL DATA UNITS

5.1.1 META-AMS PROTOCOL DATA UNIT FORMAT

The protocol data units used to effect configuration and discovery procedures are termed *Meta-AMS (MAMS)* PDUs, or *MPDUs*. Each MPDU shall consist of a header in fixed format followed by a variable-length digital signature and/or variable-length supplementary data; the length of the digital signature in an MPDU may be zero in the event that AMS is not configured for MAMS traffic security, and otherwise may be up to 255 octets; the length of the supplementary data in an MPDU shall vary from zero to 4096 octets.

The MPDU header shall consist of the fields shown in table 5-1. The MPDU header fields shall be transmitted in the order of presentation in table 5-1.

Table 5-1: MAMS PDU Header Fields

Field	Length (bits)	Values	Comment
Sender's venture number	8		Zero if sender is configuration service.
Sender's unit number	16		
Sender's role number	8		Zero if sender is not a node.
MPDU type	8	See table 5-2 below.	
Length of digital signature	8		
Length of supplementary data	16		
Memo	32	See table 5-2 below.	Semantics vary by MPDU type.
Time tag	32		As from Unix time() function.

5.1.2 META-AMS PROTOCOL DATA UNIT TYPES

The MAMS protocol data unit type shall indicate the nature of the MPDU as noted in table 5-2.

Table 5-2: MAMS PDU Types

MPDU Type (decimal)	Function
00	(reserved)
01	heartbeat
02	rejection
03	you_are_dead
04	config_msg_ack
05	are_you_active
06	(reserved)
07	announce_registrar
08	(reserved)
09	(reserved)
10	cell_spec
11	(reserved)
12	(reserved)
13	(reserved)
14	(reserved)
15	(reserved)
16	msg_space_query
17	(reserved)
18	registrar_query
19	node_registration
20	you_are_in
21	I_am_starting
22	I_am_here
23	declaration
24	subscribe
25	unsubscribe
26	I_am_stopping
27	reconnect
28	cell_status
29	(reserved)
30	(reserved)
31	node_status
32	I_am_running
33	(reserved)
34	invite
35	disinvite
All other values	(reserved)

5.1.3 META-AMS MEMO VALUES

5.1.3.1 Query Number. A query number sequentially assigned by AMS for the purposes of a synchronous MAMS message exchange. Each AMS communicating entity has its own distinct series of query numbers.

5.1.3.2 Echo. The additive inverse of the memo number of the received MPDU that caused this MPDU to be sent.

5.1.3.3 Heartbeat Source. If heartbeat is from a node, then the node's number. Otherwise, zero.

5.1.3.4 Node ID. A numeric value identifying the node to which the MPDU pertains: the sum of node number plus (256 * unit number) plus (16777216 * role number).

5.1.4 META-AMS DIGITAL SIGNATURES

5.1.4.1 General

A digital signature may be included in a MAMS message to assure the receiver of the message that the sender is authentic. The nature of the digital signature shall depend on the source and destination of the message. Each digital signature shall be generated by:

– selecting four ASCII characters – the 'nonce' ('number used once') for the digital signature – at random;

– encrypting in the private key of the sender the concatenation of the nonce and some well-known string;

– concatenating the nonce, in clear text, with the encrypted string to form the digital signature.

5.1.4.2 Node Signature

For any MAMS message sent by an AMS node, the applicable digital signature shall be generated from the node's role name using the private key of the role.

5.1.4.3 Registrar Signature

For any MAMS message sent by a registrar, the applicable digital signature shall be generated from the message space's application name using the private key of the application.

5.1.4.4 Configuration Server Signature

For any MAMS message sent by a configuration server, the digital signature shall be generated from the continuum's name using the private key of the continuum.

5.1.5 META-AMS SUPPLEMENTARY DATA VALUES

5.1.5.1 General

The term *string* is used here to signify an array of text characters formed by the concatenation of one or more lexical tokens – and/or other strings – delimited by single spaces. All string text is in ASCII representation. The last character of the last text character in a supplementary data value is immediately followed by a NULL character, terminating the string.

All binary integer values are in network byte order.

5.1.5.2 MAMS Endpoint Name

A MAMS endpoint name is a string comprising the name, within the namespace identified by the primary transport service name, of some endpoint at which MAMS messages may be received.

5.1.5.3 Cell Descriptor

A cell descriptor is the concatenation of unit number (an eight-bit integer) and the MAMS endpoint name of the cell's registrar.

5.1.5.4 Node List

A node list is the concatenation of an eight-bit integer indicating the number of nodes in the list, followed by that number of eight-bit integer node numbers.

5.1.5.5 Assigned Node Number

Assigned node number is an eight-bit integer containing the node number newly assigned to the recipient node.

5.1.5.6 Delivery Point Name

Delivery point name is the concatenation of transport service name, an equals (=) symbol, and the name (within the namespace identified by that name) of some endpoint at which the node can receive AMS PDUs.

5.1.5.7 Delivery Vector

A delivery vector is the concatenation of a four-bit integer identifying the delivery vector (the delivery vector number) and a four-bit integer indicating the number of delivery point names in the vector, followed by that number of comma-delimited delivery point names, terminated by a NULL character. These delivery point names identify a set of endpoints at which AMS messages can be delivered in the service mode that is common to all the transport services cited in the vector.

5.1.5.8 Delivery Vector List

A delivery vector list is the concatenation of an eight-bit integer indicating the number of delivery vectors in the list, followed by that number of delivery vectors. The list identifies all delivery vectors established for the node.

5.1.5.9 Subscription Assertion Structure

A subscription assertion structure is the concatenation of a 16-bit integer subject number; a 24-bit integer identifying the continuum containing all the nodes to which the asserted subscription pertains; a 16-bit integer identifying the unit containing all the nodes to which the asserted subscription pertains; an eight-bit integer identifying the role characterizing all the nodes to which the asserted subscription pertains; the four-bit delivery vector number of the delivery vector to use in transmitting messages on the subject; a four-bit integer indicating the priority at which transmission of the messages is requested; and an eight-bit flow identifier integer with which it is requested that the messages be labeled.

5.1.5.10 Invitation Assertion Structure

An invitation assertion structure is the concatenation of a 16-bit integer subject number; a 16-bit integer identifying the unit containing all the nodes to which the asserted invitation pertains; an eight-bit integer identifying the role characterizing all the nodes to which the asserted invitation pertains; the four-bit delivery vector number of the delivery vector to use in transmitting messages on the subject; a four-bit integer indicating the priority at which transmission of the messages is requested; and an eight-bit flow identifier integer with which it is requested that the messages be labeled.

5.1.5.11 Subscription List

A subscription list is the concatenation of a 16-bit integer indicating the number of subscription assertion structures in the list, followed by that number of subscription assertion structures. The list identifies subjects to which the node subscribes.

5.1.5.12 Invitation List

An invitation list is the concatenation of a 16-bit integer indicating the number of invitation assertion structures in the list, followed by that number of invitation assertion structures. The list identifies subjects to which the node does not subscribe but on which it is willing to accept messages.

5.1.5.13 Declaration Structure

A declaration structure is the concatenation of a delivery vector list, a subscription list, and an invitation list.

5.1.5.14 Node Status Structure

A node status structure is the concatenation of the node’s own MAMS endpoint name (which is NULL-terminated, as it is the last string in the supplementary data value) and a declaration structure.

5.1.5.15 Subscription Cancellation Structure

A subscription cancellation structure is the concatenation of a 16-bit integer subject number; a 24-bit integer identifying the continuum containing all the nodes to which the canceled subscription pertains; a 16-bit integer identifying the unit containing all the nodes to which the canceled subscription pertains; and an eight-bit integer identifying the role characterizing all the nodes to which the canceled subscription pertains.

5.1.5.16 Invitation Cancellation Structure

An invitation cancellation structure is the concatenation of a 16-bit integer subject number; a 16-bit integer identifying the unit containing all the nodes to which the canceled invitation pertains; and an eight-bit integer identifying the role characterizing all the nodes to which the canceled invitation pertains.

5.1.5.17 Reconnect Structure

A reconnect structure is the concatenation of an eight-bit integer indicating the node number of the node, followed by the node’s own MAMS endpoint name (which is NULL-terminated, as it is the last string in the supplementary data value), followed by the cell’s node list.

5.1.5.18 Refusal Reason

Refusal reason is an eight-bit integer code explaining the rejection: ‘1’ = duplicate registrar, ‘2’ = no cell census yet, ‘3’ = cell is full.

5.1.6 META-AMS PROTOCOL DATA UNIT STRUCTURES

The contents of MAMS protocol data units shall be as specified in table 5-3.

Table 5-3: MAMS PDU Structures

Type		Memo	Supplementary Data
01	Heartbeat	Heartbeat source	None
02	Rejection	Echo	Refusal reason
03	You_are_dead	Echo	None
04	config_msg_ack	Echo	None
05	Are_you_active	Query number	None
07	announce_registrar	0	MAMS endpoint name
10	cell_spec	Echo	Cell descriptor
16	msg_space_query	0	None
18	registrar_query	Query number	None
19	node_registration	Query number	MAMS endpoint name
20	You_are_in	Echo	Assigned node number
21	I_am_starting	Node ID	MAMS endpoint name
22	I_am_here	Node ID	Node status structure
23	declaration	Node ID	Declaration structure
24	subscribe	Node ID	Subscription assertion structure
25	unsubscribe	Node ID	Subscription cancellation structure
26	I_am_stopping	Node ID	None
27	reconnect	Query number	Reconnect structure
28	cell_status	Node ID (but node and role numbers in ID are zero)	Node list
31	node_status	Node ID	Node status structure
32	I_am_running	0	None
34	invite	Node ID	Invitation assertion structure
35	disinvite	Node ID	Invitation cancellation structure

5.2 AMS MESSAGES

5.2.1 GENERAL

Each AMS message shall consist of a header in fixed format followed by zero or more octets of content. The length of the content in an AMS message shall vary as indicated by the content length field of the header.

The AMS message header shall consist of the fields shown in table 5-4. The PDU header fields shall be transmitted in the order of presentation in table 5-4.

Table 5-4: AMS Message Header Fields

Field	Length (bits)	Values	Comment
Source continuum number	24		To support Remote AMS operations.
Source unit number	16		
Source node number	8		
Message subject number	16		
Context number	32		
Reply flag	1		'1' = the message is a reply to an antecedent message with the same context number.
spare	3		
Priority	4	1-15	Lower numeric values signify 'higher priority', i.e., greater urgency in delivery of the data.
Flow label	8		No AMS-defined semantics.
Content length	16		Limited to 65,000.

5.2.2 REPLIES

For any AMS message issued in response to a **Publish.request** primitive, a **Send.request** primitive, or a **Query.request** primitive:

– the 'reply' flag shall be set to zero;

– context number shall be non-zero if a reply is invited.

Such messages are termed 'non-reply' messages.

For any AMS message issued in response to a **Reply.request** primitive, the 'reply' flag shall be set to '1' and context number shall be equal to the context number of the message to which this reply is a response. Such messages are termed 'reply' messages.

5.3 REMOTE AMS COMMUNICATION PDUS

5.3.1 GENERAL

For the purposes of Remote AMS (RAMS) communication, AMS message traffic must be encapsulated in other data structures that enable forwarding through RAMS gateways. This encapsulation is described here.

5.3.2 REMOTE AMS PDUS

Each RPDU shall comprise a single *envelope* structure as described below.

5.3.3 ENVELOPES

The data structures used to direct the forwarding of AMS message traffic by source and destination RAMS gateways are termed *envelopes*.

Each envelope shall consist of a header in fixed format followed by zero or more octets of content; if the length of the content exceeds zero, then the content shall be an *enclosure* structure as described below. Envelopes with control code ‘2’ or ‘3’ (petition assertions and cancellations) must have zero octets of content.

The envelope header shall consist of the fields shown in table 5-5. The envelope header fields shall be transmitted in the order of presentation in table 5-5.

Table 5-5: Envelope Header Fields

Field	Length (bits)	Values	Comment
Control code	8	‘2’ = petition assertion ‘3’ = petition cancellation ‘4’ = publish on reception ‘5’ = send on reception ‘6’ = announce on reception	All other values are reserved for future use.
Continuum number	24		
Unit number	16		
Node number	8		
Role number	8		
Subject number	16		
Length of content	16		

The contents of envelope header fields shall be constrained as described in table 5-6.

Table 5-6: Envelope Header Field Contents

Control code	Continuum number	Unit number	Node number	Role number
‘2’	Identifies publisher(s)	Identifies publisher(s)	0	Identifies publisher(s)
‘3’	Identifies publisher(s)	Identifies publisher(s)	0	Identifies publisher(s)
‘4’	0	0	0	0
‘5’	Destination of message	Destination of message	Destination of message	0
‘6’	Destination of message	Destination of message	0	Destination of message

5.3.4 ENCLOSURES

An *enclosure* is a data structure that is identical in format and semantics to an AMS message, except that it is never passed to a transport service for transmission. Instead:

– Enclosures may constitute the contents of envelopes (described above) received by source and destination RAMS gateways.

– When destination RAMS gateways send, announce, or publish messages, the contents of those messages are enclosures that were received as the contents of envelopes.

– The enclosures that are the contents of messages sent, announced, or published by destination RAMS gateways and received by application nodes are, in practical effect, application messages that were sent, announced, or published by application nodes in remote continua.

6 MANAGEMENT INFORMATION BASE

6.1 NODE MIB

The MIB of each node shall include:

– The name and number of the local AMS continuum.

– The name of the Primary Transport Service.

– The Primary Transport Service endpoint names of all network locations at which the configuration server for this continuum is authorized to operate, in descending order of preference.

– The names of all transport services by which the node is able to transmit AMS messages.

– The default MADP specification.

– Standard delivery preference rules, for translating between service modes and delivery point names and specifications.

– Optionally, the AMS public encryption key characterizing the continuum, used by the configuration server.

– Optionally, the AMS public encryption keys characterizing one or more of the supported applications, used by the registrars of message space cells in which the node may register.

– Optionally, the AMS private encryption key characterizing one functional role within one supported application, used by the node in registering with that role's name.

– Optionally, a list of all functional role names under which nodes are authorized to register in any of the message spaces in which the node may register, together with the AMS public encryption key for the application node(s) that may register under each such name.

– For each message space in which the node may register, a list of all subjects, identified by both name and number, on which messages may be issued within this message space. Associated with each subject is:

  • For subject numbers greater than zero **only** ('application subjects'), subject catalog information including:

    ▪ a description of this message subject, discussing the semantics of messages issued on this subject;

    ▪ a detailed specification of the structure of the content of messages on this subject;

  - optionally, a specification of the manner in which a correctly assembled message is marshaled for network transmission in a platform-neutral manner and, on reception, un-marshaled into a format that is suitable for processing by the application.

- Optionally, security information including:
  - a list of the functional role names of all nodes that are authorized senders of messages on this subject;
  - a list of the functional role names of all nodes that are authorized receivers of messages on this subject;
  - encryption parameters, including a symmetric encryption key, enabling encryption of messages on this subject.

6.2 REGISTRAR MIB

The MIB of each registrar shall include:

- The name and number of the local AMS continuum.
- The name of the Primary Transport Service.
- The Primary Transport Service endpoint names of all network locations at which the configuration server for this continuum is authorized to operate, in descending order of preference.
- The period on which to autonomously re-advertise the configuration of the cell.
- Optionally, the AMS public encryption key characterizing the continuum, used by the configuration server.
- Optionally, the AMS public encryption key for the application characterizing the message space that contains the cell served by this registrar.
- Optionally, the AMS private encryption key for the application characterizing the message space that contains the cell served by this registrar.
- A list of all functional role names under which nodes are authorized to register in the cell served by this registrar. Associated with each role name is, optionally, the AMS public encryption key for the application node(s) that may register under that name.

6.3 CONFIGURATION SERVER MIB

The MIB of each configuration server shall include:

– The name and number of the local AMS continuum.

– The name of the Primary Transport Service.

– The Primary Transport Service endpoint names of all network locations at which the configuration server for this continuum is authorized to operate, in descending order of preference.

– A list of all applications for which message spaces may be constructed. Associated with each application name is, optionally, the AMS public encryption key for the application.

– Optionally, the AMS private encryption key characterizing the continuum, used by the configuration server.

ANNEX A

INFORMATIVE REFERENCES

(INFORMATIVE)

TBD.

ANNEX B

ACRONYMS

(Informative)

AMS	Asynchronous Message Service
DTN	Delay Tolerant Networking
FIFO	First In First Out
LAN	Local Area Network
MADP	Meta-AMS Delivery Point
MAMS	Meta-AMS
MIB	Management Information Base
MPDU	Meta-AMS PDU
MSB	Most Significant Bit
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PTS	Primary Transport Service
RAMS	Remote AMS
RPDU	RAMS PDU
SAP	Service Access Point
SDU	Service Data Unit
STS	Supplementary Transport Service

ANNEX C

RECOGNIZED TRANSPORT SERVICES

(INFORMATIVE)

tcp

TCP endpoint specifications may be represented in either of the following ways:

portnumber:hostname	
portnumber	[indicating that the hostname is the canonical name of the local host]

The transport service name for all TCP delivery points is ‘tcp’. The endpoint name for each TCP delivery point is ‘*portnumber*:*hostname*’.

Each TCP delivery point name is therefore an ASCII string of the form ‘tcp=*portnumber*:*hostname*’.

udp

UDP endpoint specifications may be represented in either of the following ways:

portnumber:hostname	
portnumber	[indicating that the hostname is the canonical name of the local host]

The transport service name for all UDP delivery points is ‘udp’. The endpoint name for each UDP delivery point is ‘*portnumber*:*hostname*’.

Each UDP delivery point name is therefore an ASCII string of the form ‘udp=*portnumber*:*hostname*’.

fifo

FIFO endpoints may not be specified; they are automatically chosen by the AMS implementation.

The transport service name for any FIFO delivery point is ‘fifo*hostnumber*’ where *hostnumber* is the 32-bit IP address of the host machine in which the FIFOs are constructed. (Note that AMS communication via FIFO is only possible between nodes residing on a common host machine.) The endpoint name of each FIFO delivery point is ‘*pathname*’.

Each FIFO delivery point name is therefore an ASCII string of the form ‘fifo*hostnumber*=*pathname*’.

<u>vxpipe</u>

VxWorks pipe endpoints may not be specified; they are automatically chosen by the AMS implementation.

The transport service name for any VxWorks pipe delivery point is ‘vxpipe*hostnumber*’ where *hostnumber* is the 32-bit IP address of the host machine in which the pipes are constructed. (Note that AMS communication via VxWorks pipe is only possible between nodes residing on a common host machine.) The endpoint name of each VxWorks pipe delivery point is ‘/pipe/*owntaskID*’.

Each VxWorks pipe delivery point name is therefore an ASCII string of the form ‘vxpipe*hostnumber*=/pipe/*owntaskID*’.

<u>vxmsgq</u>

VxWorks on-board message queue endpoints may not be specified; they are automatically chosen by the AMS implementation.

The transport service name for any VxWorks on-board message queue delivery point is ‘vxmsgq*hostnumber*’ where *hostnumber* is the 32-bit IP address of the host machine in which the message queues are constructed. (Note that AMS communication via VxWorks on-board message queue is only possible between nodes residing on a common host machine.) The endpoint name of each VxWorks on-board message queue delivery point is ‘*msgqID*’.

Each VxWorks on-board message queue delivery point name is therefore an ASCII string of the form ‘vxmsgq*hostnumber*=*msgqID*’.

<u>smmsgq</u>

VxWorks shared-memory (off-board) message queue endpoints may not be specified; they are automatically chosen by the AMS implementation.

The transport service name for any VxWorks shared-memory message queue delivery point is ‘smmsgq*busnumber*’ where *busnumber* is the administratively assigned 32-bit number (e.g., spacecraft ID) that uniquely identifies the set of processors sharing access to the memory board in which the shared-memory message queues are constructed. (Note that AMS communication via VxWorks shared-memory message queue is only possible between nodes residing on a common bus.) The endpoint name of each VxWorks shared memory message queue delivery point is ‘*msgqID*’.

Each VxWorks shared-memory message queue delivery point name is therefore an ASCII string of the form ‘smmsgq*busnumber*=*msgqID*’.

ANNEX D

SUBJECT NUMBER SIGNIFICANCE

(Informative)

The table below summarizes the functional significance of various classes of AMS subject numbers and the manner in which this significance affects protocol procedures. For nodes in continuum N (where N must be > 0), processing of messages on subject Q is as follows:

Value of Q	**Significance**	**Processing by RAMS gateway only**	**Processing by all nodes including RAMS gateway**
Q = (0 – N)	Announcements destined for nodes in *all continua* (including N).	<u>Subscribes</u> to Q at startup. On reception of a message on Q, forwards it as 'announce on reception' (with destination continuum number 0) to all neighbors.	Invisibly, <u>publishes</u> messages on Q when announcing to nodes in *all continua*. (The continuum number of the encapsulated announcement is zero).
Q < 0, Q <u>not</u> equal to (0 – N)	Private messages and announcements destined for nodes in continuum (0 – Q).	<u>Subscribes</u> to Q in response to petitions asserted by neighboring gateways (in their initial declarations or at the time they respond to the gateway's own initial declaration). On reception of a message on Q, forwards it as 'send on reception' or 'announce on reception' (with destination continuum (0 – Q)) to the neighbor that has petitioned for messages on Q.	Invisibly, <u>publishes</u> messages on Q when sending a message privately to a node in continuum (0 – Q) or when announcing only to nodes in (0 – Q).

Value of Q	**Significance**	**Processing by RAMS gateway only**	**Processing by all nodes including RAMS gateway**
Q = 0	a) Messages received from nodes in continua other than N.	a) None.	a) Invisibly, invites messages on Q at startup. On reception of a message on Q, extracts the encapsulated enclosure (which is on a subject number > 0) and processes it.
	b) Published messages on *all subjects* > 0, from nodes in the local continuum.	b) None.	b) Subscribes to Q in order to receive published traffic on *all subjects* > 0, constrained by domain. **Note** that no messages on subject Q are ever published by any node. The subscription to Q merely serves to notify AMS at each node that all published messages are subject to delivery to this subscribing node.

Value of Q	**Significance**	**Processing by RAMS gateway only**	**Processing by all nodes including RAMS gateway**
Q > 0	Published messages on subject Q.	Subscribes to Q in response to petitions asserted by neighboring gateways (as they respond to local subscriptions). On reception of a message on Q, envelopes it and forwards it as ‘publish on reception’ to each neighbor that has petitioned for messages on Q.	Subscribes to Q in order to receive published traffic on subject Q, constrained by domain.
any	all	On reception of an envelope forwarded from a neighbor, which has N or 0 as its destination continuum number, sends it privately in a message on subject 0 to every node in the message space that has subscribed to the subject of the enclosure (if ‘publish on reception’) or is the destination of the message (if ‘send on reception’) or is in the domain of the message (if ‘announce on reception’).	